Nulla di nuovo nel Sistema Solare

Giovani e intrepidi i programmatori danno il loro contributo

di Emmanuele Somma

Se quaranta anni ti sembrano tanti prova a vedere come si programmava, 40 anni fa. La sfida di scoprire e raccontare questo semplice fatto ci ha galvanizzato. Celebrare l'anniversario della missione lunare Apollo 11 è stata quindi la scusa perfetta per dare l'idea e la sensazione di questo.

Solo poco tempo prima Alan Turing aveva teorizzato che la nascita del calcolatore digitale avrebbe fatto sorgere una nuova leva di professionisti e che si sarebbe sviluppata una loro cultura e professionalità specifica. Turing realizzò il primo linguaggio e fu quindi senza dubbio il primo programmatore in senso moderno.

I giovani programmatori che lanciarono la Columbia nello spazio e contribuirono a far atterrare l'Aquila dell'Apollo 11 sulla Luna avevano di fronte sfide incredibili di una complessità che oggi spaventerebbero il più intrepido di noi. Non solo per la difficoltà del compito in sé stesso, che includeva nozioni molto avanzate di molti campi dell'ingegneria, dalla balistica alla radiotelemetria, ma perché –e per la prima volta nella storia umana– nelle circa 200 ore della missione la sopravvivenza stessa di tre persone dipese in modo determinante da come un software era stato scritto. Una responsabilità che ai giorni d'oggi sembra continuamente essere accettata, a volte senza neppure tanta consapevolezza, ma che allora senza dubbio era sconcertante.

Molti degli interpreti di questa storia erano giovani, se gli astronauti non raggiungevano i quarant'anni, gran parte dei tecnici non arrivava ai trenta, i programmatori poi avevano anche poco più di venti anni.

Ma c'era qualcuno che era ancora più giovane di loro: il computer. La tecnologia per la costruzione dei sistemi, i grossi mainframe, era già abbastanza solida e rodata. Quella tecnologia però voleva aver a disposizione una ampia stanza condizionata con grandi disponibilità elettriche, tecnici in camice bianco, schede perforate e quanto altro.

Non si mettevano a quel tempo computer in una scatola di scarpe. Quella fu una sfida nuova, incredibile e estrema.

Per il 40° anniversario, gli astronauti della missione Apollo 11, ricevuti alla Casa Bianca dal Presidente Obama sono andati a perorare la causa di una nuova missione che porti un uomo a mettere piede sul suolo rosso di Marte. Vorremmo tutti veramente vedere quel giorno. E quel giorno forse avremo il rimpianto che il software che guiderà la missione non sarà stato scritto da un beatnik capellone di 23 anni.

Oppure no: sarà stato costruito a partire da un progettino open-source creato da un liceale finlandese.

Nulla di nuovo nel Sistema Solare, quindi.

NOTA TECNICA (ovvero Le ingenuità della programmazione dei tempi d'uscita): Al momento della chiusura del numero pronto della distribuzione ci siamo accorti che... è estate inoltrata! Siamo perfettamente in sincrono rispetto ai "nostri" tempi, ma una grossa ingenuità iniziale ci ha fatto programmare l'uscita in questo momento così poco 'adeguato'. Così abbiamo deciso di spostare l'uscita della rivista agli inizi di Settembre quando la maggior parte dei lettori sarà rientrata. C'è una ragione tecnica in questo: attualmente il dato del numero di download è l'unico dato attendibile che possiamo dedurre sulla circolazione delle riviste. Non possiamo proprio permetterci di vanificare un campionamento. Abbiate pazienza.

CP/Computer Programming
è una rivista realizzata da
Infomedia Editori Coop.
e **Ass. Turing Club**

Direzione e Amministrazione:
Via Iannacchini, 28 - AV
Reg. Trib. AV al n. 8/09
(06/05/09)

Direttore Responsabile
A. Somma

Chief Editor
A. Peluso (catonano@infomedia.it)

Managing Editor
G. Spagnuolo (kirash@infomedia.it)
M. Monti (mmonti@infomedia.it)

Layout and Composizione
XED.it (info@xed.it)

Grafica
Redsign (www.redsign.it)
grafica@infomedia.it)

Autori

D. Casali
F. Bini
F. Marzano
G. Veneri
L. Colucci
M. Distratis
M. Mesenzani
M. Nielsen
M. Veneri
R. Burkey

L'impaginazione automatica di questa rivista è realizzata al 100% con strumenti Open Source usando OpenOffice, Emacs, BHL, LaTeX, Gimp, Inkscape e i linguaggi Lisp, Python e BASH

For copyright information about the contents of Computer Programming, please see the section "Copyright" at the end of each article if exists, otherwise ask authors. Infomedia contents is © 2009 Infomedia and released as Creative Commons 2.5 BY-NC-ND. Turing Club content is © 2009 Turing Club released as Creative Commons 2.5 BY-ND.

Le informazioni di copyright sul contenuto di Computer Programming sono riportate nella sezione "Copyright" alla fine di ciascun articolo o vanno richieste direttamente agli autori. Il contenuto Infomedia è © 2009 Infomedia e rilasciato con Licenza Creative Commons 2.5 BY-NC-ND. Il contenuto Turing Club è © 2009 Turing Club e rilasciato con Licenza Creative Commons 2.5 BY-ND. Si applicano tutte le norme di tutela dei marchi e dei segni distintivi.

È in ogni caso ammessa la riproduzione parziale o totale dei testi e delle immagini per scopo didattico purché vengano integralmente citati gli autori e la completa identificazione della testata.

Manoscritti e foto originali, anche se non pubblicati, non si restituiscono.

Contenuto pubblicitario inferiore al 45%.

Indice

Vol. XVII Numero 3, Luglio-Agosto 2009

EDITORIALE

Nulla di nuovo nel Sistema Solare 3

Emmanuele Somma

COVER STORY

Atterraggio lunare 7

Emmanuele Somma

Non è facile apprezzare quanto ancora oggi risulti grandiosa la sfida tecnica che ha permesso all'uomo di raggiungere la Luna. Non solo la tecnologia informatica, ed in particolare la programmazione, giocò un ruolo fondamentale ed insostituibile per la realizzazione dell'impresa ma fu la prima volta che con completa confidenza l'uomo pose la propria vita nelle mani di un computer.

Il progetto yaAGC 18

Ron Burkey

Il progetto yaAGC fornisce uno strato di software che emula l'hardware che fu impiegato allora e che equipaggiò le navi spaziali. Fornito degli stessi programmi dell'epoca (disponibili) e degli stessi segnali in input, fornirà gli stessi segnali in output. Attenzione: i questi segnali non erano fruibili solo attraverso la tastiera e il display; erano in grado di rilevare la posizione nell'orbita, la velocità, di accendere i motori e far cambiare orbita alla nave, farla virare, farla atterrare e ridecollare. La vita degli astronauti dipese da quesa soluzione. E aveva solo 4 kilobytes di memoria !

PROGRAMMING

L'ABC di jQuery 30

David Folletto Casali

Partendo da una sorta di auto-riflessione nata da un post su Internet mi è venuto in mente che possa essere utile se iniziassi una sorta di rassegna sulla libreria jQuery. Non sapete cos'è? Allora vuol dire che l'idea non era sbagliata.

USER SPACE

Monitoraggio e interattività di ambienti chiusi tramite la Stereo Vision 34

Francesco Bini, Giacomo Veneri, Marta Veneri, Luca Colucci

Si vuole descrivere un sistema low cost d'interazione fra un ambiente chiuso e le persone che ci si muovono all'interno. Utilizzando due sole webcam collegate a un computer ed algoritmi di Stereo Vision e di Motion Detection si è grado di stimare il movimento e la posizione dei soggetti in relazione a punti caldi della stanza e prevedere, di conseguenza, delle reazioni automatiche. Ad esempio si potrebbe far riprodurre un filmato quando un visitatore di un museo si avvicina ad un particolare quadro.

CODE WORLD

Traduzione statistica automatizzata 47

Michael Nielsen

Come fanno i computer a tradurre i testi da una lingua ad un'altra? I traduttori umani, per tradurre i diversi significati, tutte le sfumature, che una parola o una frase possono assumere in contesti diversi, usano una robusta dose di conoscenza sul mondo. Questo suggerisce che la traduzione automatizzata possa essere una sfida difficile, il genere di sfida che richiederebbe l'intelligenza artificiale. Eppure, sebbene i sistemi di traduzione automatica, come per esempio Google Translate o Yahoo!'s Babel Fish siano assolutamente perfettibili, riescono a fare un lavoro sorprendentememte buono. Come si spiega?

RUBRICHE

Una rivoluzione culturale in corso 54

Mina Distratis e Maurizio Mesenzani

Oggi stiamo assistendo ad una vera rivoluzione, non solo tecnologica ma anche e soprattutto sociale e culturale. In primo piano non sono più le tecnologie e gli strumenti di gestione delle informazioni ma l'appartenenza ad una o più Comunità.

FLOSS in Kosovo 56

Flavia Marzano

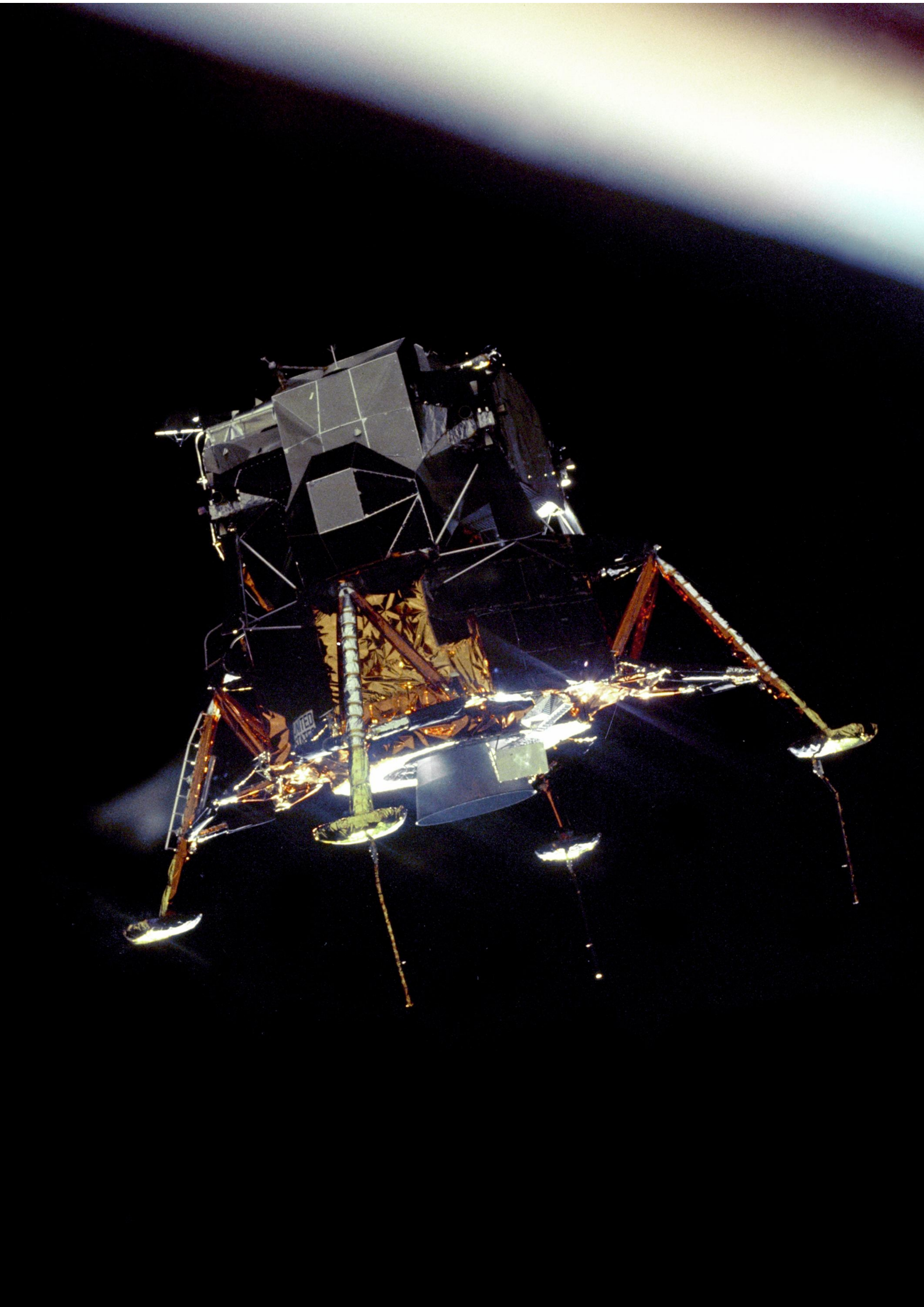
L'Università di Pristina e l'associazione FLOSSK (Free Libre Open Source Software Kosovo) hanno organizzato la prima conferenza sul Software Libero in Kosovo dal 29 al 30 agosto.

La tua FIAT in CC? 57

Giovani Spagnolo

Divertimento, passione e professionalità 58

Per collaborare con Infomedia contatta: redazione@infomedia.it



Atterraggio lunare

Come conquistare la luna in soli 4 kbyte di RAM

Emmanuele Somma

Ecco , il tuo esame di programmazione per il prossimo trimestre è questo: devi far atterrare un modulo lunare di 13 tonnellate che orbita attorno alla Luna alla velocità di 3500 kilometri all'ora, hai letto bene: 3500 km/h. Questo coso deve atterrare in una zona predefinita con non più di qualche metro di scarto. Una volta atterrato gli deve rimanere sufficiente carburante per ripartire e quindi tornare indietro, poi riprendere correttamente l'orbita e avere un rendez-vous con la nave base rimasta a girare attorno alla Luna; non hai possibilità di fare prove: tutto deve funzionare al primo colpo e soprattutto il carburante è talmente limitato che è possibile un solo tentativo di atterraggio, altrimenti si condannano i due uomini che sono all'interno a perdersi nello spazio senza che nessuno possa più recuperarli, né da vivi né da morti almeno per i prossimi trent'anni.

Hai a disposizione un computer che ha meno RAM di un VIC-20, ammesso che la si possa chiamare RAM e qualcosa come non più di 5000 primitive di programmazione a basso livello letteralmente saldate una a una.

Già perchè il circuito integrato è più o meno stato appena inventato e nessuno aveva pensato ancora di usarlo in un computer.

Non hai troppi altri vincoli però: il tutto non può pesare più di 30kg, quando un computer del tempo pesa venti volte tanto; non avrai nulla di paragonabile ad un disco su cui registrare un bel niente e alla fin fine non potrai spendere più di un paio di milioni di dollari a soldi di oggi tutto incluso, che a pensarci non è in fondo una gran cifra per l'impresa.

Ah... dimenticavo, è pure la fine degli anni '60, sei nel bel mezzo della Guerra Fredda e quindi puoi aspettarti che i tuoi arci-nemici russi, pur di farti fallire in diretta televisiva con tutto il mondo, ti disturbino le comunicazioni, le affondino in una marmellata di segnali o peggio ancora ti spediscano false informazioni di navigazione. Ai tempi si usava così... Quindi il sistema di guida dovrà pure essere autonomo e non attendersi aiuto dalla Terra: "Huston abbiamo un problema! Huston? Huston ci siete? — Da! Tovarish!"

Insomma, chi lo vorrebbe fare un programma così?

Stark Draper, responsabile del MIT Instrumentation Lab, decise di saltarci dentro e mise a disposizione del progetto spaziale il suo team di esperti per lo sviluppo dell'Apollo Computer, che divenne la base dell'AGS, l'Apollo Guidance and Navigation System, praticamente il sistema che rappresentava la più futuristica frontiera tecnologica degli anni '60. Il software fu affidato a Peter Adler e Don Eyles, quello della fase di discesa del modulo lunare ad Allan Klumpp [1].

L'Apollo Computer (AC) fu il primo ad utilizzare la tecnologia dei circuiti integrati, un grande passo per un sistema di guida ma anche un grande passo per la tecnologia da cui discenderanno direttamente i computer come siamo oggi abituati a conoscere.

Vista la loro perizia come piloti, gli astronauti avrebbero più che volentieri guidato la missione manualmente ma, come sottolineò in una lezione l'astronauta David

Gli stemmi di tutte le missioni Apollo

Il momento del lancio del vettore della missione Apollo 11

Scott: "se la Terra fosse una palla da basket e la Luna una pallina da tennis messe a 4 metri di distanza l'una dall'altro, allora il corridoio da percorrere per tornare a casa sarebbe rappresentato dallo spessore di un sottile foglio di carta che unisce le due" [1]. Senza l'AGS non sarebbe mai stato possibile garantire in un periodo d'operazioni così lungo come un viaggio Terra-Luna l'accuratezza nel controllo della nave e nella navigazione necessaria a mandare gli astronauti sulla Luna e farli tornare vivi e vegeti sulla Terra in tutta sicurezza, in modo indipendente da un sistema di guida basato sulla Terra.

Architettura

L'AC era diviso in due moduli: uno nel Modulo di Comando e l'altro nell'Aquila, come gli astronauti chiamavano il Modulo Lunare ed era posizionato proprio sotto l'accesso che gli uomini dovevano usare per scendere sulla superficie.

La memoria cui il sistema aveva accesso era di due tipi. Bhé... possiamo dire in termini moderni RAM e ROM, solo che la ROM era costruita come un intreccio di nuclei di ferrite a corrente coincidente in un tessuto di fili di rame racchiusi in una camicia di plastica. Il programma era stato letteralmente *cucito* a mano sui nuclei di ferrite secondo l'orientamento dei bit. Infatti il nucleo di ferrite funzionava come una sorta di piccolo trasformatore, era raggiunto da al più 64 fili. Se un filo s'innestava all'interno del nucleo allora ai capi di quel filo si sarebbe potuto leggere un '1', se l'avesse saltato sarebbe stato uno '0'. In caso di errore di programmazione si sarebbe dovuto sfilare l'intreccio e provvedere a ricucire correttamente le istruzioni dopo averle trasformate in binario e quindi ricostruito i pattern di filatura attraverso i nuclei di ferrite.

Tra anellini di ferro e fili di rame intrecciati ad uno ad

Una foto di gruppo del laboratorio del MIT coinvolto nella realizzazione del software per il progetto Apollo 11. In prima fila Vince Megna, Doc Charles Stark Draper, Don Eyles, Dave Moore, Tony Cook. In seconda fila: Phil Felleman, Larry Berman, Allan Klumpp, Bob Werner, Robert Lones, Sam Drake

uno si raggiungeva l'incredibile cifra di 36.864 termini o parole di programma, ciascuna delle quali a 15 bit. Quindi qualcosa di simile a 74 kilobyte odierni di ROM.

La memoria cancellabile, non si può veramente parlare di RAM in questo caso, era costruita in modo simile ma conteneva un piccolo elemento magnetico che poteva essere ruotato elettricamente determinando così lo stato acceso/spento del bit di memoria. Ce n'erano solo 2048 locazioni da 15 bit ciascuna.

Il computer aveva un "ciclo di clock" di 11,7 microsecondi. Tuttavia quasi tutte le operazioni di CPU richiedevano almeno 2 cicli di clock quindi si può considerare la reale velocitò di battimento come 23,4 micro-secondi.

Il microprocessore era quind scandalosamente lento in termini attuali. Girava all'equivalente di 0,043MHz. Ogni paragone con un moderno netbook o un cellulare di ultima generazione è quantomeno improvvido. Ma tanto per dire il Mac su cui sto scrivendo quest'articolo ha un processore CISC dual-core con una velocità di clock 60.000 volte maggiore. Moore ha lavorato bene! Anche il singolo MHz del mio primo Osborne One di vent'anni fa voleva dir qualcosa.

I numeri erano rappresentati utilizzando due parole di 14-bit in doppia precisione (cioè, 28 bit). Il bit 15 e 16 rappresentavano rispettivamente segno e parità per la verifica di sincronizzazione con il clock principale.

La Terra vista dalla superficie lunare

Tutti i calcoli venivano fatti in virgola fissa e non in floating-point.

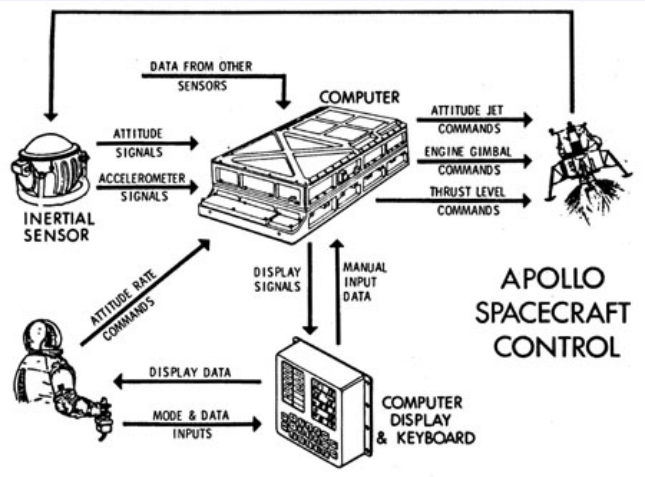
L'input e l'output dal sistema, oltre che da tutti i sensori e gli strumenti telemetrici, era garantito attraverso il *disky*, ovvero la console Display and Keyboard Unit (il codice ufficiale era però `DSKY`). Da sola pesava dieci chili ed aveva poco più di un tastierino numerico e una serie di tasti speciali per introdurre direttamente le parole e i comandi nel programma, nonchè un pannello led per leggere i principali valori del sistema (vedi Figura 2).

Il sistema operativo

Con tutte queste limitazioni i programmatori del MIT riuscirono però a metterci dentro un sistema operativo real-time e il sistema applicativo, chiamato Luminary. Il tutto era fault-tolerant, cioè in grado da riprendersi da eventuali errori, una caratteristica che fu alla base del successo della missione.

Il programma aveva un concetto di multi tasking strettamente basato sulle interruzioni temporizzate per poter trattare i task a maggiore priorità risolti in tempo definito. Inoltre gestiva una coda a priorità dei task residui in modo da poter sempre scegliere il lavoro più urgente da trattare.

Il modello del PGNS, il Sistema di Guida e Navigazione Primaria

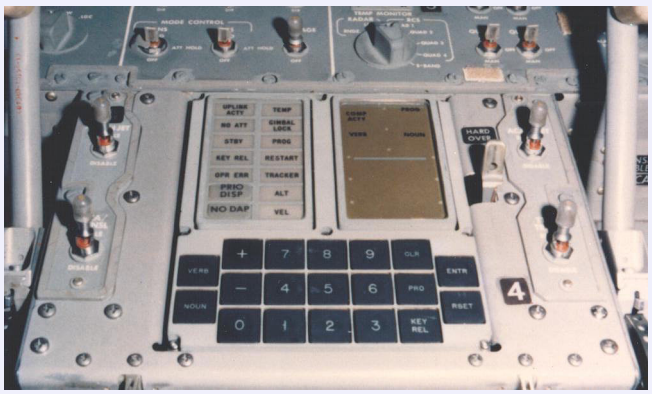


La forte limitazione della memoria cancellabile portava la stessa zona di memoria ad essere usata in modo totalmente differente a seconda del task in corso. Quando Peter Adler spiegò il funzionamento del programma dichiarò candidamente che la stessa locazione contenente ad esempio l'altitudine al suolo lunare, poteva in una fase successiva aver contenuto la lettura del sestante di una stella per l'allineamenteo della navigazione proveniente dal programma di allineamento. "Alcune locazioni furono condivise fino a sette volte in fasi diverse e si può ben immaginare i test che abbiamo dovuto fare per assicurarci che la stessa locazione di memoria non fosse utilizzata da più di un sottoprogramma alla volta nella stessa fase".

Caratteristico dell'AC era il fatto di mantenere in memoria le informazioni fondamentali per l'esecuzione dei task anche se fosse stato necessario fare un reboot a seguito di un crash del sistema.

Il sistema scelto nell'AC (priority-interrupt) impone sempre la messa in esecuzione dei lavori a più alta priorità interrompendo quelli a minore priorità, a differenza dei sistemi a time-slicing che dedicano sempre un po' di tempo a tutti i task in esecuzione indipendentemente dalla loro importanza, così che le *fettine* di tempo dedicate ai processi siano in sostanza ininterrompibili.

L'unità DSKY, che gli astronauti chiamavano affettuosamente disky era la console di input-output del Modulo Lunare. (photo credit: NASA)



I due programmi che gestivano la schedulazione del controllo dell'Apollo erano il programma esecutivo (Executive) le la lista d'attesa (Waitlist).

La Waitlist poteva gestire fino a nove task brevi con tempi di esecuzione minori di quattro millisecondi, task più lunghi passavano nell'Executive che ne gestiva solo fino a sette. Con un ciclo di 20 secondi l'Executive controllava la lista a priorità dei lavori e sostituiva il task in esecuzione con uno più importante se disponibile. In questo modo i task critici entravano sempre in esecuzione in tempi certi anche se il sistema avesse dovuto andare in fault o necessitasse di un reboot.

Per il debugging, i programmatori del MIT avevano avuto a disposizione un mainframe IBM 360 modello 175 programmato come un simulatore di modulo lunare. Allan Klumpp e suoi colleghi poterono testare il software su questo simulatore, che forniva gli stimoli al programma come un vero modulo lunare, anche se l'output non erano altro che stampe e schede perforate. Nel vero modulo lunare, il computer di bordo aveva un display digitale e una tastiera. Durante l'atterraggio il

display emetteva una serie di informazioni. Il Pilota, che era a destra del display e aveva l'ordine di non toccare il computer, avrebbe solo dovuto continuamente leggere i valori sul display. Il Comandante, alla sinistra del DSKY, avrebbe invece manipolato i controlli, ma poteva vedere le informazioni sul display solo riflesse sulla finestra d'uscita del modulo. Da questa finestra era visibile anche il punto d'atterraggio in modo che il Comandante potesse spostarlo in caso di necessità, cosa che in effetti avvenne quando Armstrong decise che non era il caso di rischiare di atterrare in un cratere dove sembravano esserci roccie della dimensione di un'automobile. Per questa modifica del piano di volo il modulo atterrò quando erano rimasti meno di 30 secondi di carburante.

Dei due comandi manuali il primo spostava il punto d'atterraggio di un grado nelle varie direzioni, il secondo permetteva di variare la velocità d'atterraggio di un piede al secondo.

Il centro di controllo di Huston

Huston, abbiamo un problema!

Non tutto funzionava come doveva. Neil Armstrong e Buzz Aldrin, i due uomini nell'Aquila erano a seimila piedi dalla superficie, più o meno a metà strada quando si accese una luce gialla di attenzione. Era un errore 1202 che indicava un sovraccarico di memoria. Certamente una situazione imprevista che si presentava ai loro occhi con la semplice e drastica semplificazione di una luce gialla e un numero d'errore. Non è che in quei casi uno può portarsi dietro un manuale o le millesettecento pagine di programma per capire cos'è che non va!

Cosa avrebbero dovuto fare? Era necessario modificare qualcosa nel piano di volo? Eventualmente abortire la missione? Oppure dovevano semplicemente trascurare l'errore considerando il malfunzionamento ininfluente? Bastava tentare le procedure di reboot?

I due astronauti intrappolati nell'Aquila chiamarono il Controllo Missione. La decisione si spostò quindi su Steve Bales, l'esperto informatico del centro di controllo responsabile per il sottosistema del modulo lunare. Fu un altro tecnico, Jack Garman che supportava Bales da un'altra console che rieseguì mentalmente le ore e ore di simulazioni effettuate e riconobbe la situazione d'errore tra le tante che avevano già trattato in precedenza. Ne parlò con Bales che considerata la questione assicurò gli astronauti che potevano continuare. Per giungere alla decisione definitiva si impiegò un tempo che Bales, ricordando quegli attimi in una successiva intervista, non esitò a considerare lunghissimo: quindici secondi, quando "durante la discesa, nel Centro di Controllo, qualsiasi cosa oltre i tre secondi era considerato troppo lento". Bales fu a quel punto irremovibile: l'errore andava trascurato! Sì ripresentò numerose volte in seguito ma non impedì all'Aquila di completare il suo viaggio verso la superficie. Gli astronauti quindi impostarono sul *disky* la procedura di reset.

Peter Adler aveva lavorato duramente sulla procedura di reboot e preteso test estensivi della capacità di reinizializzazione del computer, una abilità che risultò fondamentale nell'occasione anche in presenza di un errore imprevisto nel trattamento dei dati.

Si scoprì poi che il sovraccarico era dovuto infatti da un flusso di dati inatteso proveniente dal puntamento radar. I task a maggiore priorità prendevano il sopravvento rispetto al flusso che però dal canto suo riempiva inopportunamente la memoria residua. Il che scatenava un'eccezione nella gestione degli errori che gli astronauti erano chiamati ad affrontare manualmente.

Il reboot permise la reinizializzazione del computer e

alla ripartenza i task in macchina venivano rieseguiti a partire dal più urgente, proprio come previsto. Ogni tanto però riapparivano gli errori 1201 o 1202 richiedendo un ulteriore reboot che reinizializzava la coda e rimetteva in esecuzione i task critici come la guida di discesa o il controllo d'assetto.

Bug a bordo

Secondo Allan Klumpp il programma di gestione del modulo lunare ma anche il computer di comando non sono mai stati bug-free, sebbene vennero usati per altre missioni dopo quella dell'Apollo 11. In particolare sono noti due dettagli degli errori residui in Luminary. Uno riguardò proprio la discesa dell'Apollo 11 raccontata in precedenza.

Il programma di discesa era stato realizzato in una prima versione proprio da Klumpp e reagiva alle richieste degli astronauti con un breve ritardo dell'ordine di 3 decimi. In questo tempo il sistema attivava delle subroutine per stimare in modo accurato il nuovo livello della spinta dei motori che peraltro richiedeva la lettura del *delta-V* (variazione di velocità) misurata dagli accelerometri del modulo lunare. L'accensione del motore non poteva essere immediata così Klumpp progettò una rutine che compensava il ritardo. Alla fine la routine di Klumpp funzionava stabilmente con un ritardo costante di 3 decimi qualsivoglia fosse il livello di spinta richiesto.

La routine finale di controllo della spinta però fu scritta da Don Eyles che entusiasticamente aveva trovato un modo per riscriverla ottenendo un ritardo di soli due decimi, e il simulatore gli dette ragione. La routine di Eyles fu montata a bordo degli Apollo 11 e 12.

Ma, come abbiamo visto, la telemetria trasmessa durante gli atterraggi provò che qualcosa andava veramente storto. I motori si sovraccaricavano e rimanevano instabili. Alla fin fine il ritardo dovuto al motore routine era molto minore dei tre decimi previsti, continui miglioramenti l'avevano portato addirittura a 7 centesimi e mezzo. Miglioramenti che semplicemente non eranto presi in considerazione dalla routine di Eyeles e quindi lasciavano il sistema in uno stato instabile.

Klumpp introdusse il problema nell'IBM 360 che faceva da simulatore e replicò il problema che si era presentato durante la discesa del modulo sulla Luna. Ora tutto quadrava: anche il simulatore con la routine di Eyles andava in sovraccarico. Ciò che Klumpp però scoprì fu veramente incredibile. Se il programma di Eyles si fosse comportato secondo le specifiche compensando il ritardo dei motori rigidamente per i 3 decimi che erano stati dichiarati dai dati di targa l'Apollo 11 non avrebbe mai potuto trovare un modo per atterrare intero sulla Luna. Per un intuito inspiegabile Don Eyles era stato così creativo da scrivere una routine che adattava non rigidamente il modulo lunare alla finestra di stabilità necessaria a farlo atterrare.

Il secondo problema noto, e insoluto fino al momento della dismissione del programma Luminary, riguardava una subroutine scritta da Klumpp stesso denominata `P64`. Questo sottoprogramma valutava periodicamente una funzione polinomiale per descrivere la traiettoria di discesa ottimale.

Il risultato della polinomiale avrebbe interpolato la posizione corrente del modulo lunare durante la discesa e, prendendo in considerazione i vettori della velocità e il punto previsto d'atterraggio e la velocità al suolo, avrebbe ottimizzato la spinta necessaria.

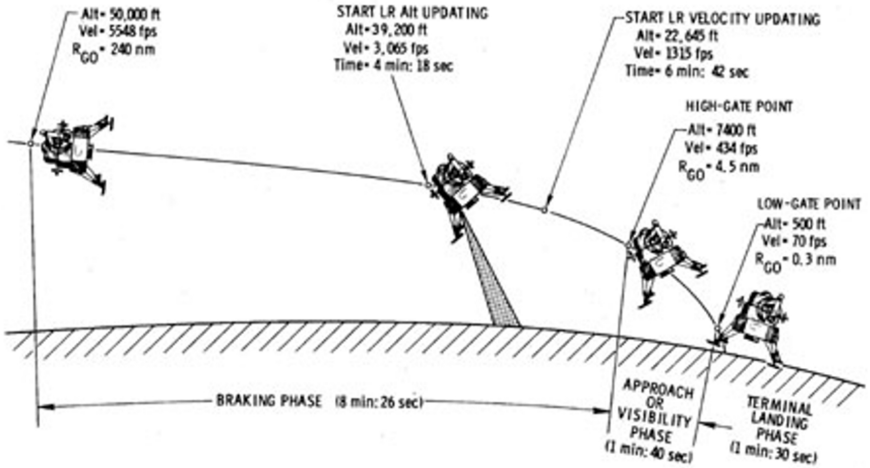
In effetti la routine `P64` avrebbe funzionato solo fino a pochi metri dal suolo quando sarebbe stata sostituita dalla routine `P66` che si occupava della breve discesa verticale fino a toccare il suolo lunare.

Era noto a tutti che la routine `P64` aveva una possibilità d'errore, un bug noto, che peraltro avrebbe potuto portare al completo disastro della missione.

Come è noto una curva polinomiale di ordine alto può presentare delle 'anche' abbastanza pronunciate. Non sarebbe stato impossibile per `P64` calcolare una funzione di questo tipo che avrebbe sì ottimizzato la funzione al punto d'atterraggio ma facendola passare da un minimo in precedenza inferiore alla stessa superficie lunare.

Il modulo, così, per seguire questa traiettoria avrebbe rischiato di impattare la Luna, senza speranza quindi di poter poi risalire e raggiungere il punto d'atterraggio. Ad esempio se il modulo lunare si fosse trovato a sorvolare un cratere abbastanza profondo da far sbagliare il radar di atterraggio era abbastanza probabile che la traiettoria calcolata non avrebbe tenuto in conto la reale presenza della superficie lunare. Insomma c'era pericolo. E restava! Infatti la routine non fu

Le fasi dell'atterraggio del modulo lunare



mai aggiustata. È sempre rimasta, fino al suo pensionamento questa possibilità, che era ben più probabile che teorica. Klumpp cercava di preparare gli astronauti insegnandogli dei trucchi per venir fuori da una possibile situazione del genere spingendo l'obiettivo d'allunaggio oltre l'intervallo permesso dal carburante, per poi ritornarvi quando il polinomio si fosse stabilizzato fuori dal pericolo.

La soluzione però, c'era. E tutti gli astronauti la conoscevano bene: il grande pulsante 'ABORT' che imponeva il termine prematuro della missione. La fase di discesa del modulo lunare sarebbe stata sganciata, sarebbero stati riaccesi automaticamente i motori della fase di risalita e si sarebbe fatto ritorno al modulo di comando senza allunare. Game Over.

Conclusioni

Don Eyles aveva 23 anni quando scrisse il programma di atterraggio del modulo lunare, si considerava un beatnik controcorrente, come si può facilmente intuire dalle sue foto dell'epoca. Oggi è un affermato artista a Boston.

Steve Bales invece aveva 26 anni e, per la sua decisione di non interrompere la missione, gli fu tributata la US Medal of Freedom al pari dei tre astronauti che avevano fatto il viaggio sull'Apollo 11.

Il discorso che Richard Nixon pronunciò a Los Angeles il 13 Agosto del '69 per celebrare il rientro della missione fu emblematico:

> Tutti coloro che hanno visitato Huston o chi ha avuto la possibilità di vistare le altre installazione del nostro programma spaziale è rimasto enormemente impressionato dagli uomini e dalle donne che vi lavorano. Siamo impressionati dalla loro intelligenza, dalla loro dedizione, e quando sono stato a Huston io stesso sono stato impressionato dalla loro gioventù
>
> L'uomo che è stato selezionato per ricevere stasera questo premio a nome dei quattrocentomila che, in un modo o nell'altro, hanno contribuito al successo di questo programma è un giovane di 26 anni. Ma Steve Bales, che era Ingegnere di Controllo del Volo di questo progetto, ha dovuto predere una decisione critica proprio prima che l'Aquila Uno atterrasse nel Mare della Tranquillità. Una decisione che avrebbe potuto fare la differenza tra il successo e il fallimento.

Don Eyles, il capellone del gruppo, e il decano Doc Charles Stark Draper

Bales, faccia un passo avanti per ricevere questo Group Achievment Award in rappresentanza di tutti quelli che dalla Terra hanno reso possibile questa avventura sulla Luna.

Ecco: questo è il giovane che quando il computer sembrava essere confuso e quando avrebbe potuto dire "Stop" o quando avrebbe potuto dire "Aspetta", disse "Vai!"

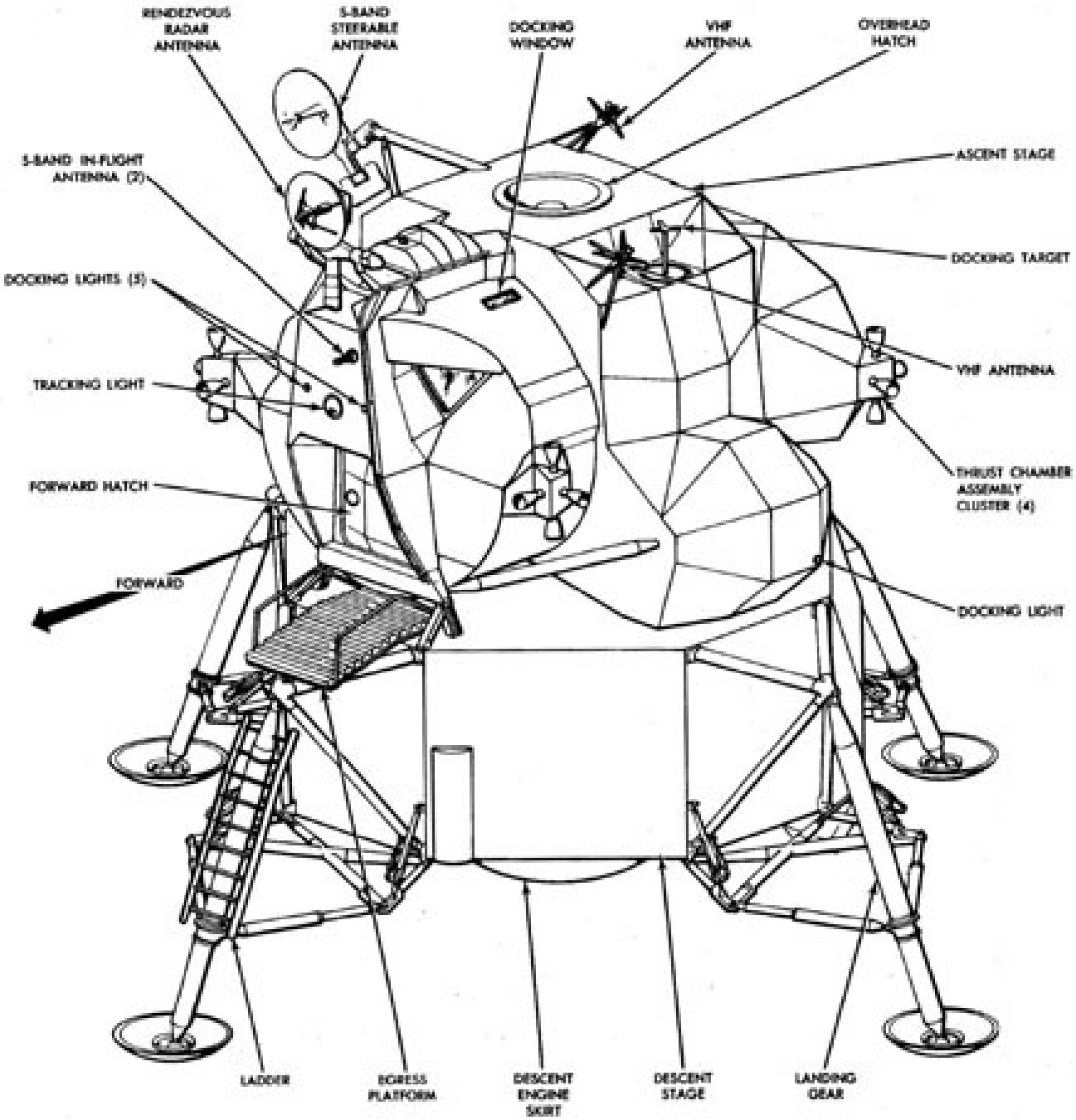
Una lezione, anche questa, che dovremmo ricordare anche ai giorni nostri.

Riferimenti

1. My Fascinating Interview with Allan Klumpp di Stanley R. Mohler, Jr. (Dec '94) in sci.space.tech
2. Apollo Expeditions to the Moon, edited by Edgar M. Cortright, NASA SP; 350, Washington, DC, 1975 Apollo Lunar Surface Journal www.hq.nasa.gov/office/pao/History/alsj
3. Richard Nxon in onore degli astronauti dell'Apollo 11. August 13, 1969 www.presidency.ucsb.edu

XKCD

Schema del modulo lunare



Listato 1 (parte 1 di 3): la routine per la ricerca delle radici del polinomio realizzata da Allan Klump

```
# ********************************************************************************
# DOUBLE PRECISION ROOT FINDER SUBROUTINE (BY ALLAN KLUMPP)
# ********************************************************************************
#
# N N-1
# ROOTPSRS FINDS ONE ROOT OF THE POWER SERIES A X + A X + ... + A X + A
# N N-1 1 0
# USING NETON'S METHOD STARTING WITH AN INITIAL GUESS FOR THE ROOT. THE ENTERING DATA MUST BE AS FOLLOWS:
# A SP LOC-3 ADRES FOR REFERENCING PWR COF TABL
# L SP N-1 N IS THE DEGREE OF THE POWER SERIES
# MPAC DP X INITIAL GUESS FOR ROOT
#
# LOC-2N DP A(0)
# ...
# LOC DP A(N)
# LOC+2 SP PRECROOT PREC RQD OF ROOT (AS FRACT OF 1ST GUESS)
#
# Page 823
# THE DP RESULT IS LEFT IN MPAC UPON EXIT, AND A SP COUNT OF THE ITERATIONS TO CONVERGENCE IS LEFT
# IN MPAC+2.
# RETURN IS NORMALLY TO LOC(TC ROOTPSRS)+3. IF ROOTPSRS FAILS TO CONVERGE TO IN 8 PASSES, RETURN IS
# TO LOC+1 AND OUTPUTS ARE NOT TO BE TRUSTED.
#
# PRECAUTION: ROOTPSRS MAKES NO CHECKS FOR OVERFLOW OR FOR IMPROPER USAGE. IMPROPER USAGE COULD
# PRECLUDE CONVERGENCE OR REQUIRE EXCESSIVE ITERATIONS. AS A SPECIFIC EXAMPLE, ROOTPSRS FORMS A
# DERIVATIVE COEFFICIENT TABLE BY MULTIPLYINE EACH A(I) BY I, WHERE I RANGES FROM 1 TO N. IF AN
# ELEMENT OF THE DERIVATIVE COEFFICIENT TABLE = 1 OR >1 IN MAGNITUDE, ONLY THE EXCESS IS RETAINED.
# ROOTPSRS MAY CONVERGE ON THE COREECT
# ROOT NONETHELESS, BUT IT MAY TAKE AN EXCESSIVE NUMBER OF ITERATIONS. THEREFORE THE USER SHOULD
# RECOGNIZE:
# 1. USER'S RESPONSIBILITY TO ASSUR THAT I X A(I) < 1 IN MAGNITUDE FOR ALL I.
# 2. USER'S RESPONSIBILITY TO ASSURE OVERFLOW WILL NOT OCCUR IN EVALUTATING EITHER THE RESIDUAL OR
# THE DERIVATIVE POWER SERIES. THIS OVERFLOW WOULD BE PRODUCED BY SUBROUTINE POWRSERS, CALLED BY
# ROOTPSRS, AND MIGHT NOT PRECLUDE EVENTUAL CONVERGENCE.
# 3. AT PRESENT, ERASABLE LOCATIONS ARE RESERVED ONLY FOR N UP TO 5. AN N IN EXCESS OF 5 WILL
# PRODUCE CHAOS. ALL ERASABLES USED BY ROOTPSRS ARE UNSWITCHED LOCATED IN THE REGION FROM MPAC-33 OCT
# TO MPAC+7.
# 4. THE ITERATION COUNT RETURNED IN MPAC+2 MAY BE USED TO DETECT ABNORMAL PERFORMANCE.

# STORE ENTERING DATA, INITIALIZE ERASABLES
ROOTPSRS EXTEND
QXCH RETROOT # RETURN ADRES
TS PWRPTR # PWR TABLE POINTER
DXCH MPAC +3 # PWR TABLE ADRES, N-1
CA DERTABLL
TS DERPTR # DER TABL POINTER
TS MPAC +5 # DER TABL ADRES
CCS MPAC +4 # NO POWER SERIES DEGREE 1 OR LESS
TS MPAC +6 # N-2
CA ZERO # MODE USED AS ITERATION COUNTER. MODE
TS MODE # MUST BE POS SO ABS WON'T COMP MPAC+3 ETC.

# COMPUTE CRITERION TO STOP ITERATING
EXTEND
DCA MPAC # FETCH ROOT GUESS, KEEPING IT IN MPAC
DXCH ROOTPS # AND IN ROOTPS
INDEX MPAC +3 # PWR TABLE ADRES
CA 5 # PRECROOT TO A
TC SHORTMP # YIELDS DP PRODUCT IN MPAC
TC USPRCADR
CADR ABS # YIELDS ABVAL OF CRITERION ON DX IN MPAC
DXCH MPAC
DXCH DXCRIT # CRITERION
```

Listato 1 (parte 2 di 3): la routine per la ricerca delle radici del polinomio realizzata da Allan Klump

```
# SET UP DER COF TABL
# Page 824
EXTEND
INDEX PWRPTR
DCA 3
DXCH MPAC # A(N) TO MPAC

CA MPAC +4 # N-1 TO A

DERCLOOP TS PWRCNT # LOOP COUNTER
AD ONE
TC DMPNSUB # YIELDS DERCOF = I X A(I) IN MPAC
EXTEND
INDEX PWRPTR
DCA 1
DXCH MPAC # (I-1) TO MPAC, FETCHING DERCOF
INDEX DERPTR
DXCH 3 # DERCOF TO DER TABLE
CS TWO
ADS PWRPTR # DECREMENT PWR POINTER
CS TWO
ADS DERPTR # DECREMENT DER POINTER
CCS PWRCNT
TCF DERCLOOP


# CONVERGE ON ROOT
ROOTLOOP EXTEND
DCA ROOTPS # FETCH CURRENT ROOT
DXCH MPAC # LEAVE IN MPAC
EXTEND
DCA MPAC +5 # LOAD A, L WITH DER TABL ADRES, N-2
TC POWRSERS # YIELDS DERIVATIVE IN MPAC

EXTEND
DCA ROOTPS
DXCH MPAC # CURRENT ROOT TO MPAC,
          # FETCHING DERIVATIVE
DXCH BUF  # LEAVE DERIVATIVE IN BUF AS DIVISOR
EXTEND
DCA MPAC +3 # LOAD A, L WITH PWR TABL ADRES, N-1
TC POWRSERS # YIELDS RESIDUAL IN MPAC
```

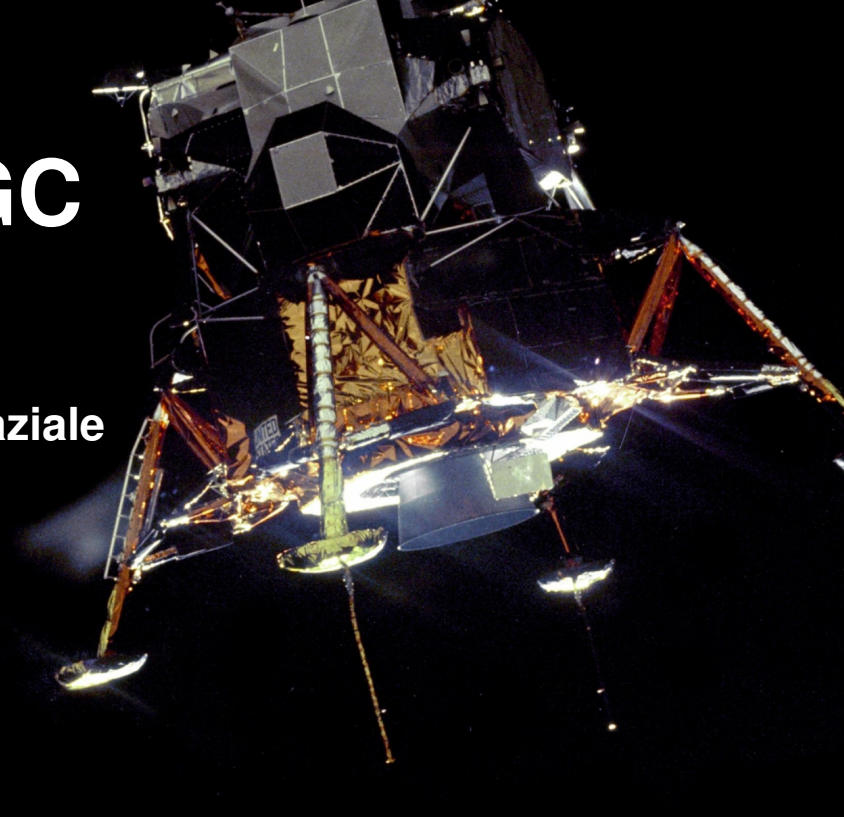
Listato 1 (parte 3 di 3): la routine per la ricerca delle radici del polinomio realizzata da Allan Klump

```


TC USPRCADR
CADR DDV/BDDV # YIELDS -DX IN MPAC


EXTEND
DCS MPAC # FETCH DX, LEAVING -DX IN MPAC
DAS ROOTPS # CORRECTED ROOT NOW IN ROOTPS


TC USPRCADR
CADR ABS # YIELDS ABS(DX) IN MPAC
EXTEND
# Page 825
DCS DXCRIT
DAS MPAC # ABS(DX)-ABS(DXCRIT) IN MPAC


CA MODE
MASK BIT4 # KLUMPP SAYS GIVE UP AFTER EIGHT PASSES
CCS A
BADROOT TC RETROOT


INCR MODE # INCREMENT ITERATION COUNTER
CCS MPAC # TEST HI ORDER DX
TCF ROOTLOOP
TCF TESTLODX
TCF ROOTSTOR
TESTLODX CCS MPAC +1 # TEST LO ORDER DX
TCF ROOTLOOP
TCF ROOTSTOR
TCF ROOTSTOR
ROOTSTOR DXCH ROOTPS
DXCH MPAC
CA MODE
TS MPAC +2 # STORE SP ITERATION COUNT IN MPAC+2
INDEX RETROOT
TCF 2


DERTABLL ADRES DERCOFN -3
```

Il progetto yaAGC

Un emulatore di hardware... spaziale

Ron Burkey



La nave del progetto Apollo usata per le missioni lunari nei tardi anni 60 e primi anni 70 era costituita in realtà da due parti diverse, chiamate *moduli*. Erano il modulo di comando e il modulo lunare.

Il modulo di comando (detto CM da Command Module, vedi **Figura 1**) era impiegato per trasportare tre astronauti in orbita attorno alla luna e farli tornare. Il modulo lunare (detto LM da Lunar Module, vedi **Figura 2**) era usato per trasportare due astronauti sulla superficie lunare mentre il terzo attendeva a bordo del modulo di comando che rimaneva in orbita.

Un po' come si fa con le grandi imbarcazioni quando si arriva in prossimità di una spiaggia: si ormeggia al largo e si raggiunge la riva con una scialuppa.

Ognuno dei moduli doveva essere in grado di navigare attraverso lo spazio, sia con, ma anche senza, il presidio degli astronauti, quindi doveva essere dotato di un *sistema di guida*. Tale sistema fu sviluppato dal Laboratorio di Strumentazione del MIT. Oggi è una azienda distinta dal MIT chiamata Charles Stark Draper Laboratory.

Una parte importante del sistema di guida era il suo computer, detto Apollo Guidance Computer, per gli amici AGC.

> Il computer di bordo dell'Apollo 11 aveva meno di 4Kb di RAM e pesava solo poco meno di 32 Kg.

Per ogni missione Apollo c'erano due AGC, uno per il modulo di comando e uno per il modulo lunare. I due erano del tutto identici ed intercambiabili, la sola differenza era nel software che vi girava; dovevano infatti svolgere compiti differenti.

Inoltre, il software che girava su quelle macchine migliorò nel tempo, quindi la versione che ci girò per le missioni Apollo più avanzate, ad esempio la Apollo 17, fu in qualche misura differente da quello che girò per le prime missioni, come la Apollo 8.

Secondo gli standard attuali, l'hardware dell'AGC era drammaticamente sottodimensionato. Le caratteristiche fondamentali erano:

- una RAM di 2048 parole. Una parola era una sequenza di 15 bit di dati —-quindi meno di 2 bytes (16 bits)-— dunque l'ammontare totale di RAM era di soli 3840 bytes, meno cioè di 4kb;
- 36,864 parole di memoria a sola lettura, equivalenti a 69,120 bytes, poco più di 64kb;

Figura 1: Il modulo di comando (photo credit: NASA)

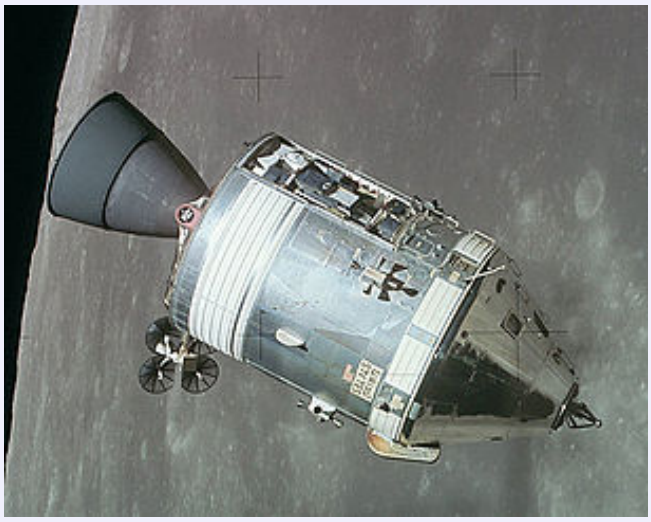


Figura 2: Il modulo lunare (photo credit: NASA)

- Un massimo di circa 85,000 istruzioni eseguite al secondo;
- Dimensioni: 24×12.5×6” (60x30x15 cm);
- Peso: 70.1 pounds (31.75 kg);
- Alimentazione elettrica: 2.5A di corrente a 28V DC

A volte viene detto —forse senza troppa consapevolezza— che l’AGC assomigliava più ad una calcolatrice che ad un computer, ma dire questo significa sottovalutare grossolanamente il livello di sofisticatezza dell’AGC. Per dirne una: era multitasking e poteva eseguire più programmi contemporaneamente.

Un’altra parte importante del sistema era l’unità che conteneva il display e la tastiera, chiamata DSKY rappresentata in **Figura 3**. L’AGC era di fatto solo una scatola con connessioni elettriche senza nessuna possibilità di interazione da parte degli astronauti.

Il modulo lunare aveva un solo DSKY, posizionato fra i due astronauti, dove lo potevano manovrare entrambi. Il modulo di comando, invece, aveva due DSKY; uno era quello principale e l’altro era collocato vicino alla attrezzatura ottica che veniva impiegata per tenere traccia della posizione delle stelle o di altri punti di riferimento del panorama.

Il DSKY era una parte di equipaggiamento autonoma:

- Dimensioni: 8×8×7” (20x20x18 cm)
- Peso: 17.5 pounds (8 Kg)

Forse la parte più importante del sistema di guida era l’unità di misura inerziale (detta IMU da Inertial Measurement Unit). La IMU teneva traccia costantemente della accelerazione e della rotazione del veicolo spaziale e riportava questi dati all’AGC.

Processando questi dati secondo opportune formule matematiche, l’AGC poteva sapere momento per momento l’orientamento e la posizione nello spazio del veicolo.

Introduzione all’AGC virtuale

Il progetto *AGC virtuale* fornisce una macchina virtuale che emula l’AGC, il DSKY e alcune altre parti del

Figura 3: Il modulo DSKY che incorporava tastiera e display (photo credit: NASA)

sistema di guida. In altre parole, se a questa macchina virtuale —che noi chiameremo yaAGC, ovvero *yet another AGC*— viene fornito lo stesso software che girava sugli AGC di allora e se le viene fornito lo stesso flusso di segnali in input che veniva fornito agli AGC di allora durante le missioni Apollo, allora questa macchina fornirà le stesse risposte che avrebbero fornito gli AGC di allora.

Attenzione, però: questo software emula la CPU di allora, un po' come gli emulatori delle consolle da videogiochi o come gli emulatori del retro computing tipo quello che emula il Commodore 64.

Questo software *non è* un simulatore di volo o di allunaggio e non è un simulatore del comportamento del modulo lunare o di quello di comando.

Quindi se vi aspettate che un pannello di controllo realistico appaia sul vostro monitor rimarrete un po' delusi.

Ripetiamo, emula una CPU. Per ricreare una simulazione più realistica, bisognerebbe emulare anche altre parti fondamentali dei sistema di guida, quali la strumenazione ottica, l'unità di misura inerziale, il pannello DSKY, gli stessi motori dei moduli.

Questo software potrebbe però fornire una base valida per la realizzazione di una simulazione più completa. Il Virtual AGC è gratuito ed è disponibile in una versione per Windows, una per Mac Os X, una per Linux, oppure come base di codice con licenza GPL così che possa essere studiato e modificato.

Il progetto AGC virtuale fornisce una macchina virtuale che emula l'AGC, il DSKY e alcune altre parti del sistema di guida

Cosa fornisce esattamente queso progetto

Gli elementi di AGC sono:

- yaAGC è un emulatore della CPU dell'AGC. Per funzionare richiede un file binario che emuli a sua volta un modulo lunare o un modulo di comando.
- yaDSKY è una simulazione del pannello che conteneva la tastiera e il display. Fornisce l'input allo yaAGC, o riceve l'output da esso. Il design è modulare, quindi il pezzo di software che simula il DSKY può facilmente essere rimpiazzato da un altro, implementato magari da voi stessi.
- yaTelemetry è un terminale stupido dal quale possono essere visti i collegamenti di telemetria dallo yaAGC con la stazione di controllo a terra, in modo del tutto simile a quello che accadde in realtà nel Controllo Missione.
- yaACA è uno strato software che simula la cloche del modulo lunare che potrebbe essere rappresentata efficacemente da un joy stick. La cloche era naturalmente collegata all'AGC e gli comunicava i cambiamenti desiderati di pitch/roll/yaw.
- yaYUL è un assembler, in grado di convertire il codice assembly AGC4 in un binario eseguibile dall'AGC.

- Luminary è il nome del programma usato dall'AGC del modulo lunare delle missioni Apollo. Sono forniti sia una versione binaria eseguibile dall'AGC sia una versione in codice sorgente (che può essere digerita dallo yaYUL).

- Colossus è il nome del programma usato per molti (ma non tutti) i moduli di comando delle missioni Apollo. Come per Luminary, ci sono sia la versione in codice sorgente che quella binaria.

- Validation è il nome di un programma scritto in assembly per AGC creato in era contemporanea che fornisce una suite di validazione che aiuta ad assicurarsi che l'implementazione della CPU dell'AGC sia corretta. E' stato scritto seguendo strettamente la documentazione originale delle missioni Apollo che descrive la suite di validazione impiegata allora.

- La documentazione del programma Apollo rilevante ai fini della programmazione dell'AGC, che fra l'altro è stata corretta, riscritta e in alcuni casi espansa - nel caso che vogliate scrivere i vostri programmi per AGC. Vengono fornite anche le scansioni dei documenti originali dell'epoca non disponibili altrove.

- Un simulatore di un modulo lunare, che è stato scritto da Stephan Hotto e che non è più una parte aggiuntiva ma viene distribuito quale parte integrante del progetto. Fornisce

  - una simulazione della Unità di Misura Inerziale (IMU)
  - una sfera FAI
  - un DSKY diverso da yaDSKY che può essere usato se per qualche ragione yaDSKY dovesse essere una opzione non praticabile
  - un monitor dei canali di input/output

  Per congiungere tutto insieme c'è *VirtualAGC* che è un front end che fornisce una GUI su tutto questo; il tutto dovrebbe rendere l'emulazione "facilmente" accessibile (sul perché *facilmente* sia fra virgolette non ci dilunghiamo).

Far girare l'emulatore . . . ovvero, una breve introduzione alla GUI

Il primo passo, naturalmente, è quello di scaricare il software e installarlo (o farne la *build*) in base della piattaforma su cui si opera, secondo le istruzioni sulla pagina di download (www.ibiblio.org/apollo/download.html).

Mentre il progetto fornisce programmi diversi ognuno dei quali implementa una parte diversa della simulazione, l'unico programma di cui dovrete preoccuparvi è quello chiamato *VirtualAGC*.

VirtualAGC è un front end che fornisce una GUI che consente di scegliere quasi tutte le combinazioni di opzioni ragionevoli che l'emulatore offre e quindi di far girare l'emulazione con quel set di opzioni.

L'inseme dei programmi che gireranno e il modo in cui comunicheranno fra di loro, in base alle opzioni che avrete scelto, dovrebbero risultarvi trasparente.

La mia speranza è che VirtualAGC risulti sufficientemente semplice da non richiedere troppe spiegazioni per essere usato e che i problemi comincino solo dopo che l'emulazione sia partita! Ma fornirò qui una breve panoramica.

La schermata riportata in **Figura 4** mostra la sola finestra di VirtualAGC. Non ci sono menù, barre di strumenti o altri controlli.

Benché la quantità di opzioni sia vasta non è necessario che cambiate i valori di default, se premete semplicemente il bottone "Run!" in basso, la simulazione inizierà a girare.

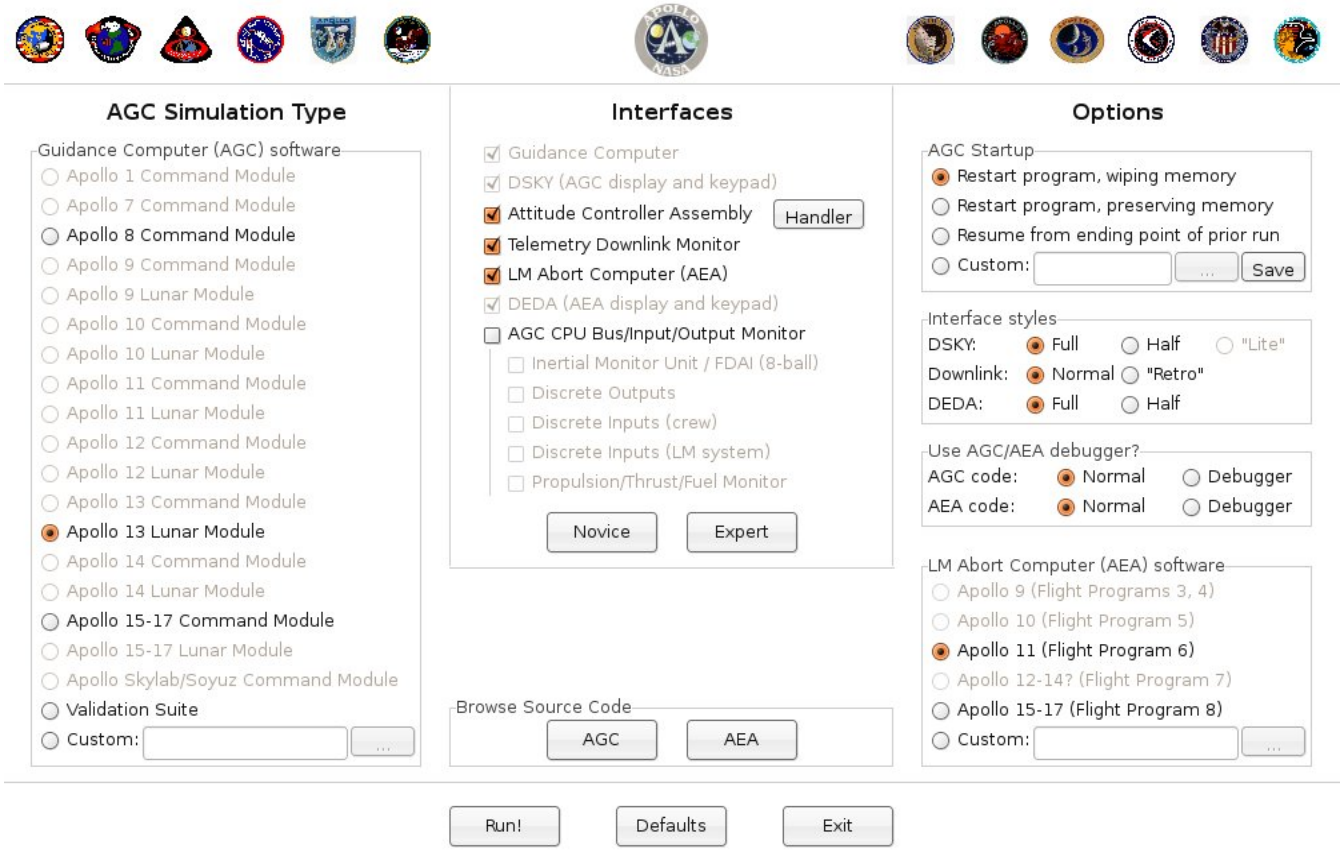
Se cambiate qualche settaggio, il programma ricorderà i vostri cambiamenti e quando lo lancerete la volta successiva si ripresentarà come lo avevate lasciato.

Naturalmente, premendo il bottone "defaults" saranno ristabiliti i valori di default.

Per un principiante c'è ben poco che possa essere cambiato in modo indolore ma noterete che potete scegliere fra le diverse versioni del software che furono impiegate nelle diverse missioni Apollo nel pannello etichettato "Simulation Type".

Mentre la simulazione gira, la finestra principale sarà tanto cortese da scomparire, per risparmiare spazio sul monitor, e sarà sostituita da una finestrella più piccola che conterrà i pochi controlli che servono.

Figura 4: VirtualAGC — la schermata di partenza (photo credit: NASA)



Chiudendo uno qualunque dei componenti che verranno aperti per la simulazione si chiuderanno anche tutti gli altri, anche se ci sono alcune speicificità delle singole piattaforme, per cui suggerisco di chiudere il DSKY.

Attenzione, la chiusura a cascata delle altre finestre comincia a funzionare solo dopo alcuni secondi che la simulazione è partita (per lo più 5 secondi, ma in alcun versioni 30) e su Mac Os ci sono alcuni programmi che da soli proprio non si chiudono (Whishe Terminator) per cui dovrete chiuderli a mano.

Quick Start

Immagino che chiunque voglia usare l'AGC virtuale abbia intenzione di adattarlo un poco, quindi le istruzioni che mi accingo a dare potrebbero non essere d'aiuto. Per non parlare del peso di dover imparare le specificità dell'intero sistema degli Apollo per poter davvero *usare* l'AGC!

Tuttavia ci sono alcune cose che si possono far accadere, benché possano risultare di poco senso a noi poveri esseri limitati ai confini della Terra!

Naturalmente la *experience* piena delle navi Apollo non è alla portata perché mancano i pezzi che simulino alcuni pezzi di hardware come l'unità di misura inerziale e l'unità ottica, che non sono state ancora creati. Ma ci sono alcune cosette che saranno apprezzate dai geek ;-)

Lanciare la suite di validazione della CPU

C'è una suite di validazione che ho scritto nell'assembly dell'AGC che controlla che ogni istruzione dell'instruction set sia implementata correttamente dall'emulatore. Non è proprio l'*Apollo experience* ma almeno saprete che state eseguendo codice assembler per quella CPU!

1. Lanciate VirtualAGC, selezionate la "Validation Suite" nell'area "Simulation Type", e premete il bottone "Run!".

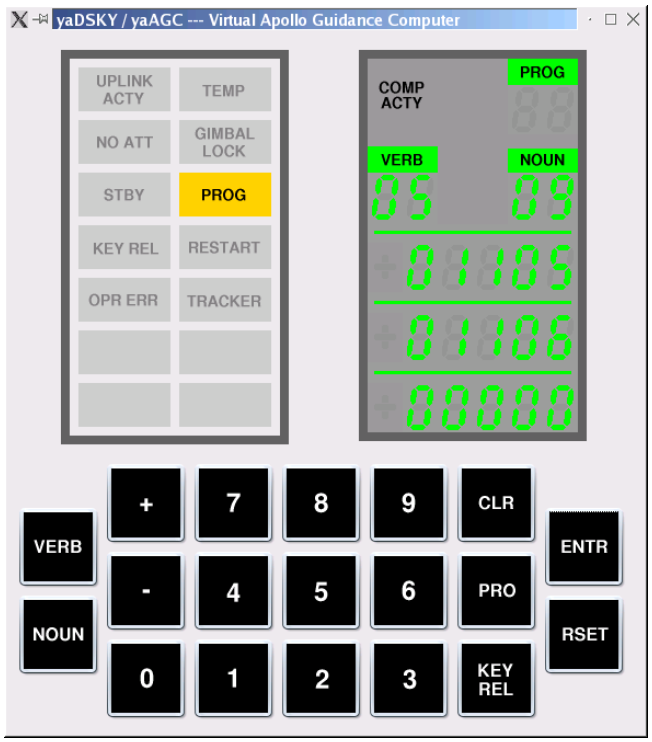
2. Sul DSKY, vedrete apparire 00 nei campi PROG e NOUN, e vedrete l'annunciatore OPR ERR illuminarsi. Ciò significa che il programma di validazione è pronto ad essere avviato. Per avviarlo premete il tasto PRO del DSKY. L'annunciatore OPR ERR si spegnerà per indicare che il comando è stato accettato.

3. Dopo circa 77 secondi, apparirà il numero 77 nel campo PROG e l'annunciatore OPR ERR si illuminerà di nuovo. Se nel campo PROG comparirà qualunque numero diverso da 77, allora quei numeri nei campi PROG e NOUN indicheranno quale test sia fallito (Potete premere di nuovo PRO per procedere con gli altri test) .

Giocare con Colossus

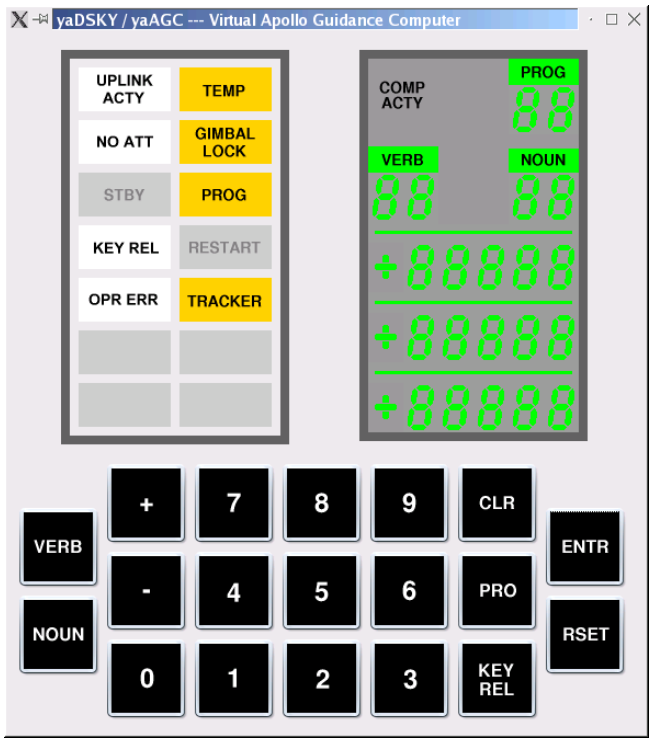
In realtà tutto quanto funziona con Luminary, funziona anche con Colossus.
Lanciate VirtualAGC, scegliete uno qualunque dei moduli di comando come "Simulation Type" e fate queste prove:

Vedere i codici d'allarme



A causa di alcuni bug nel modo in cui si inizializza Colossus allo startup ci saranno dei codici di allarme e l'indicatore PROG si illuminerà per indicarlo. Potete vedere quali siano i codici premendo `V05N09E` sul DSKY ('V' è l'abbreviazione per 'VERB' e 'N' è l'abbreviazione per 'NOUN', 'E' sta per 'ENTER' quindi 'V05N09E' indica la sequenza di tasti `VERB 0 5 NOUN 0 9 ENTER`).
I codici d'allarme saranno 1105 e 1106 che significano "downlink troppo veloce" e "uplink troppo veloce". Downlink e Uplink erano le denominazioni per i collegamenti telemetrici con la stazione di terra.

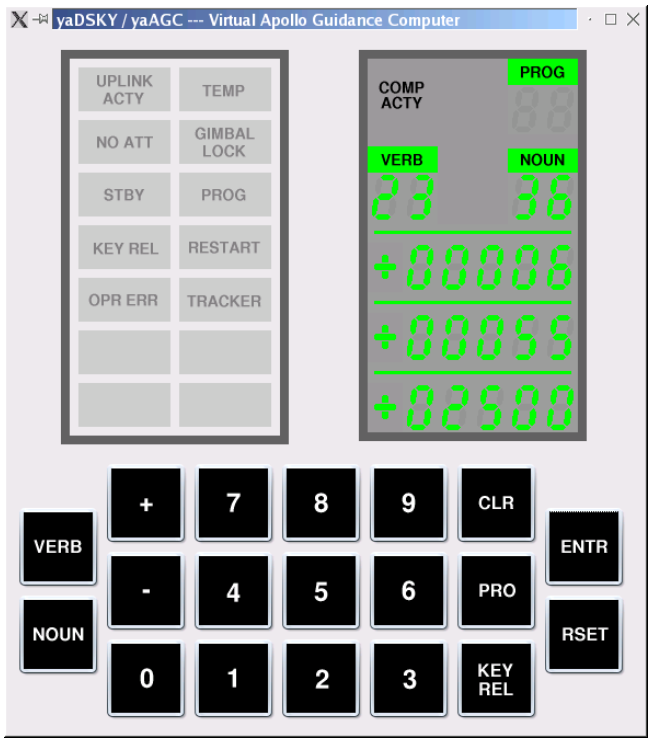
Test delle luci del DSKY



Sul DSKY premere la sequenza V35E. Questo farà illuminare tutti gli annunciatori del pannello farà lampeggiare le etichette VERB/NOUN e mostrerà 88 o +88888 su tutti i registri numerici. Dopo 5 secondi il test si conclude - ve ne accorgerete perché smette di lampeggiare, anche se i numeri rimangono - e potrete continuare.

Quando è stato fatto questo screenshot ancora non sapevo che l'AGC controlla anche gli indicatori del DSKY chiamati STBY e RESTART quindi in quel test quelli ancora non si accendevano. A causa di un bug (al 19 luglio 2004) l'indicatore PROG non si riillumina dopo il test, quindi potreste vederlo illuminato o no, se provate questa manovra.

Definire l'orario



Se vi secca di vedere l'ora a partire dal momento dell'accensione, potete settare l'orario (ad esempio a quello della missione) inserendo la sequenza 'V25N36E'. R1 si svuoterà consentendovi di digitare il valore dell'orario. Assicuratevi di iniziare la sequenza con un + (per dire all'AGC che l'orario è espresso col sistema decimale e non con l'ottale) e premete tutte e cinque le cifre, anche gli zeri. In caso d'errore, potete resettare R1 premendo CLR.

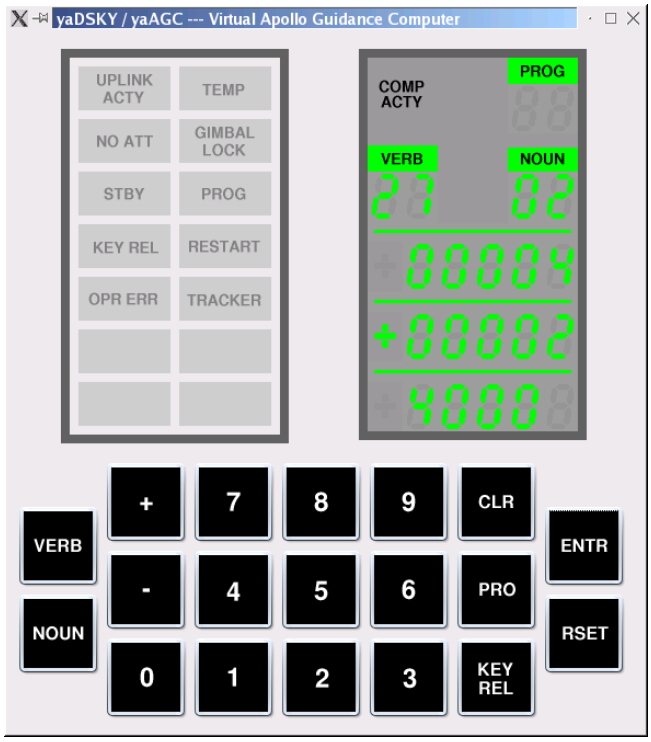
Quando avete finito con R1 premete ENTR e passerete a R2.

Infine inserirete i secondi in R3.

Non dimenticate che il numero di secondi è in centesimi di secondo quindi se ad esempio volete inserire il valore di 30 secondi dovrete battere +03000E

Nella immagine a corredo capitava che fossero le 06:55:25 di mattina e così quello è l'orario che ho inserito

Esaminare la memoria read-only

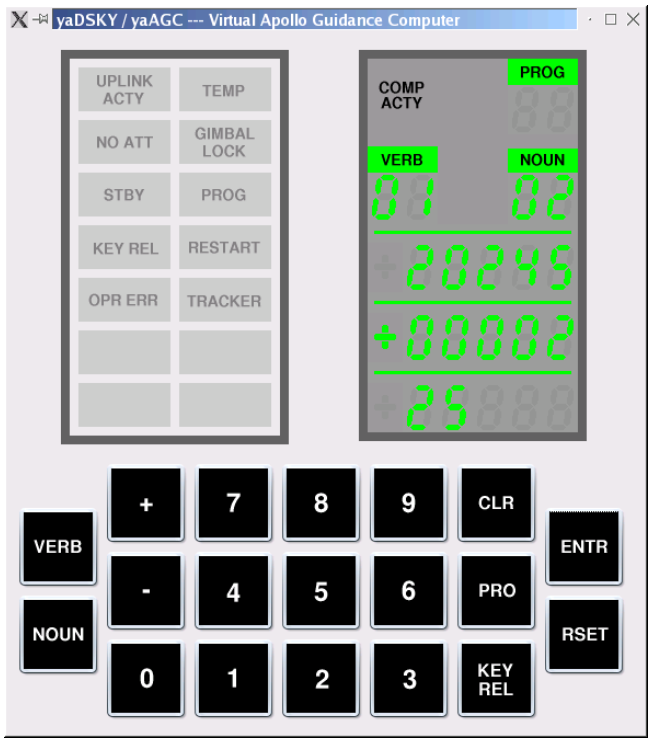


Digitate V27N02E. Questo vi consentirà di inserire in R3 l'indirizzo di una parola del core-rope. Tale indirizzo generalmente sarà in forma ottale e quindi non preceduto da un segno +. Inoltre, doversamente da quanto accade coi decimali, in ottale potete inserire quante cifre volete e non siete tenuti ad inserirne almeno 5. Gli indirizzi saranno `00000-01777` per il banco di memoria `00`, `02000-03777` per il banco `01` e così via fino a `76000-77777` per il banco `37` (non sono sicuro su come si possa fare per esaminare i banchi `40-43`).

Il listato in binario del core-rope è nella estremità posteriore del listato di Colossus 249, che può essere scaricato dal sito del MIT, se avete tempo e spazio disco.

Nello screenshot a corredo, vediamo che l'indirizzo `4000` (ottale) del core-rope di Luminary contiene il valore `00004`. Questo valore è la prima istruzione eseguita dopo l'accensione. E' una istruzione INHINT e disabilita gli interrupt. I contenuti di R2 (il display a 5 cifre centrale) non sono stati resettati, quindi mostrano quel che c'era prima. ??? linger ???

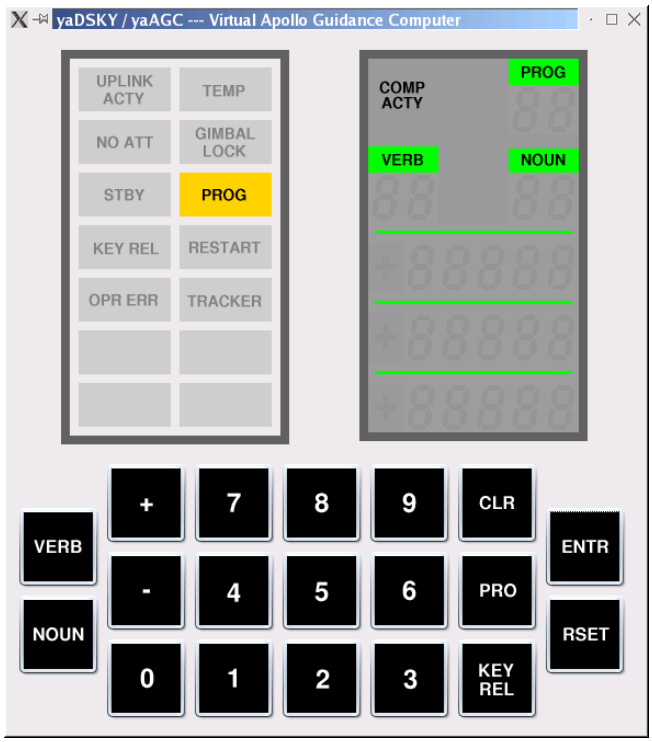
Esaminare la memoria scrivibile



Digitate V01N02E. Ciò vi consentirà di inserire l'indirizzo di una locazione di memoria scrivibile in R3. Gli indirizzi saranno `00000-00377` per il banco scrivibile E0, `00400-00777` per il banco E1 e così via fino a `03400-03777` per il banco E7. Alternativamente, potete *monitorare* una locazione di memoria, cioé ottenere il valore che contiene aggiornato ogni secondo, usando il VERB 11 invece dello 01. Per esempio, V11N02E25E monitorerà il registro 25, chiamato registro "TIME 1", che è un contatore interno che si incrementa ogni 10 millisecondi. In generale, naturalmente, i numeri non significheranno granché se non consultate il listato in assembler di Colossus 249.

Nello screenshot a corredo, stiamo vedendo il registro TIME 1 e scopriamo che in quell'istante conteneva il valore 20245 (ottale). Naturalmente, voi vedrete qualcos'altro. R2 non è cambiato, quindi conterrà qualunque cosa avesso contenuto in precedenza.

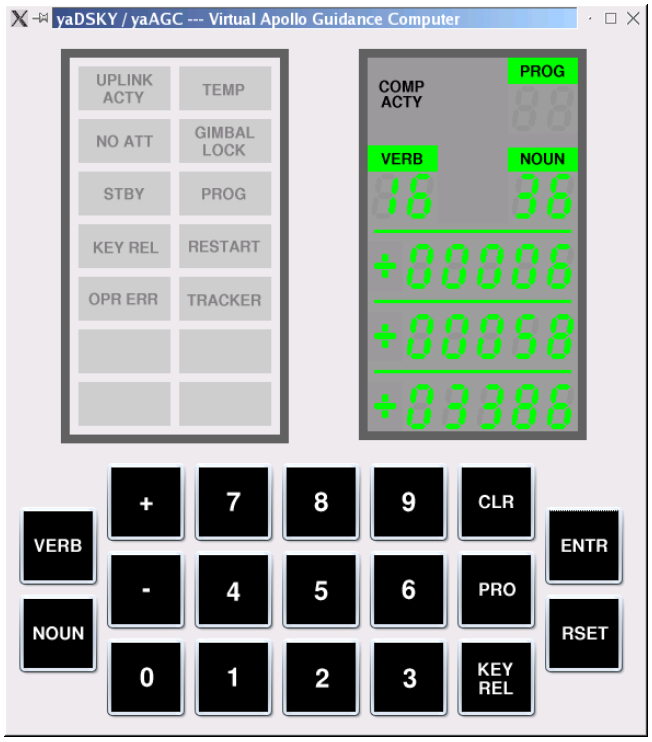
Reboot



Digitate V36E. Questo riavvia il programma pinball - cioé il programma repsonsabile di accettare verbi e sostantivi (VERBs and NOUNs) e mostrare la roba sui display del DSKY (più o meno come il Finder del Mac Os X o Esplora Risorse di Windows) - ed è utile per resettare i display del DSKY, come mostra lo screenshot a corredo.

In questo screenshot, un effetto collaterale della partenza da zero è la totale riscrittura sui display dell'output di PROG (programma di allarme); i display erano stati precedentemente cancellati dal test delle luci.

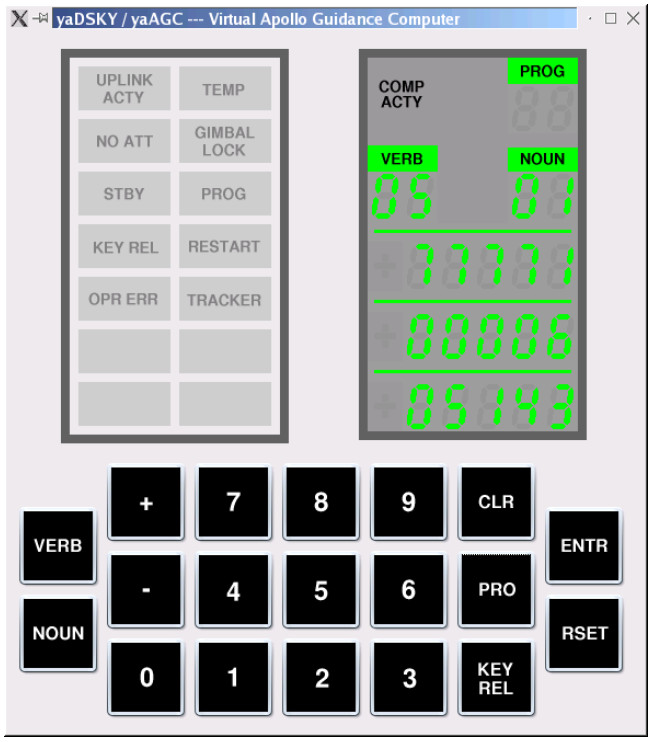
Vedere il tempo corrente



Le sequenze 'V16N36E' oppure 'V16N65E' faranno si che venga mostrato il tempo attuale (siccome non avremo settato nulla, il tempo attuale sarà quello trascorso a partire dall'accensione dell'AGC). R1 (il display a 5 cifre) mostrerà il tempo in ore, R2 in minuti, R3 sarà in centesimi di secondo. I valori saranno aggiornati una volta al secondo

In questa immagine, l'ora mostrata è 06:58:33.86

Checksum dei banchi di memoria



La memoria a sola lettura è divisa in 36 banchi numerati da 00 a 43 (ottale); una cosiddetta *parola di test* è stata piazzata alla fine di ogni banco il che fa si che il checksum di ogni banco risulti essere un valore noto. Tale valore è il numero del banco, quando è possibile, altrimenti è il complemento logico del numero del banco (ad esempio il checksum del banco 00007 è 00007, mentre quello del banco 00006 è 77771, entrambi sono corretti). Il programma "mostra il checksum dei banchi" di Colossus può essere impiegato per mostrare i numeri dei banchi uno ad uno e può essere invocato premendo la sequenza 'V91E' sul DSKY

Dopo alcuni secondi, saranno mostrati i risultati del banco 00: R1 (il display a 5 cifre) mostrerà il checksum calcolato; R2 mostrerà il numero del banco ed R3 mostrerà la parola di test

Ognuno dei display sarà in modalità ottale e la cosa è indicata dal fatto che l'indicatore +/- sarà vuoto.

Per passare al banco successivo premete la sequenza 'V33E' (oppure premete PRO).

Se avrete la pazienza di avanzare di banco in banco fino all'ultimo, quando sarete al banco 43 (l'ultimo) PRO o la sequenza di cui sopra vi faranno tornare al banco 00.

Per uscire dal test dei banchi premete la sequenza 'V34E'.

La parola di test mostrata per il banco 6 (05143) è per Colossus, se fosse stato Artemis 072 sarebbe stata 04275, per Luminary 131 sarebbe stata 63402

Le vostre ricerche, qui

La lista delle tabelle dei NOUNs e dei VERBs è contenuta nel file:

```
yaAGC/Colossus249/ASSEMBLY_AND_OPERATION_INFORMATION.s
```

In questo modo potete scoprire qualcosa di simpatico voi stessi. Se ci riuscite, fatecelo sapere, lo agguingeremo a questa lista.

Giocare con l'assembly dell'AGC

Potete modificare la suite di validazione che fornisco io e poi far girare la vostra versione modificata, facendo come segue:

1. Fate una copia della cartella che contiene la suite stessa (nello snapshot per gli sviluppatori, sono i files *yaAGC/Validation/*.s*; nella versione binaria, sono sempre i files *.s*, ma il modo più facile di trovare la cartella in cui si trovano è di "Navigare il Codice Sorgente" dall'interno di VirtualAGC e annotare qual'è la cartella che viene referenziata);

2. Editare alcuni dei file della suite. Per l'assembly fate riferimento al manuale in PDF;

3. Lanciate VirtualAGC, selezionate "Custom" nell'area "Simulation Type", e scegliete *Validation.s* nella cartella in cui state lavorando. Dopo che VirtualAGC avrà fatto girare automaticamente l'assembly sulla vostra cartella per produrre un binario eseguibile, premete semplicemente il bottone "Run!".

4. Dovreste vedere il DSKY fare ciò per cui lo avete programmato.

5. Nel caso che il vostro programma non faccia ciò che vi aspettavate, c'è una funzionalità di base di debugging all'interno di yaAGC che potrebbe esservi d'aiuto per ricostruire quale sia il problema. Nell'area "Options" di VirtualAGC, sotto "AGC Debug" scegliete "Run with debug monitor"

Per i coraggiosi: modificare il modulo che simula il modulo lunare

È la stessa cosa che con la suite di validazione, ma partite dal codice di Luminary 131 invece che dalla suite e considerate che il file *custom* che dovete selezionare è *MAIN.s* invece che *Validation.s*.

Esame finale (per studenti avanzati)

> Nella missione Apollo 14, prima della discesa del modulo lunare verso la superficie del nostro satellite, un difetto nel pannello DSKY causò l'attivazione intermittente dello switch che indicava il fallimento della discesa. Se questo fosse accaduto durante la discesa, la missione sarebbe automaticamente andata in modalità "fallimento" e questo avrebbe causato il fatto che la parte inferiore del modulo sarebbe stata eiettata e quella superiore sarebbe stata sparata nello spazio. L'allunaggio non sarebbe stato possibile e gli astronauti avrebbero fronteggiato la grave situazione di dover essere soccorsi dal modulo di comando.
>
> Fu quindi necessario, un paio di orbitazioni prima della discesa, che i programmatori dell'AGC inventassero un rimedio che consentisse di isolare lo switch "fallimento" via software nella fase preliminare della discesa e riattivarlo in una fase più avanzata, perché gli astronauti avrebbero allora potuto averne bisogno.
>
> Il rimedio fu trovato.

La vostra missione, se doveste accettarla, consiste nel trovare lo stesso rimedio voi stessi, in circa 90 minuti, poi mandatecelo. Ricordate che la soluzione può coinvolgere solo la memoria scrivibile perché la parte contenente il programma non poteva essere alterata via software e la cosa doveva poter essere inserita nella tastiera dagli astronauti. Pronti? Via!

...oppure, per i pigroni

Potete usare il VirtualAGC come orologio per la vostra scrivania (accidenti se è versatile! Parola mia!). Suggerisco di usare, come "DSKY Type" quello "Half-Size". Usate e istruzioni date in precedenza su come settare l'orario. Dovrete resettare l'orario la mattina presto perché l'indicatore delle ore non tornerà a 0 dopo aver raggiunto il 24 (e comunque il contatore andrà in overflow dopo 228 centesimi di secondo o dopo 31 giorni).
Suggerimento: una volta che avrete sistemato questa cosa, probabilmente desidererete poterla avviare automaticamente senza dover avviare VirtualAGC ogni volta. Ebbene, quando VirtualAGC fa girare una simulazione lo fa per mezzo di file batch sotto Windows e per mezzo di script di shell sotto Unix. Dalla finestra di VirtualAGC col bottone "More" vi sarà mostrato il sorgente di tali script. Partite da quello per modificarlo o estenderlo per avviare la vostra simulazione personalizzata.

...o, per i pigri che più pigri non si può

Scrivete il vostro programma per AGC che implementi una calcolatrice o un gioco tipo il tic tac toe o qualche altra cosa frivola. Sarà pure blasfemo, ma preparerà le vostre competenze per quando dovesse uscire il prossimo annuncio di lavoro per programmatori di AGC

Soluzione al problema "Esame Finale" per la missione Apollo 14

Precedentemente, in maniera scherzosa solo a metà, abbiamo proposto come problema "esame finale" una situazione che si presentò veramente durante il tentativo di Alan Shepard e Edgar Mitchell di allunare, con il loro modulo lunare, durante la missione Apollo 14.
Paul Fjeld ha fornito dettagli in merito sull' Apollo Lunar Surface Journal, usando come fonte i verbali della missione.

Il nostro Onno Hommes ha ottenuto una spiegazione direttamente da Don Eyles, che fu lo sviluppatore per AGC che risolse il vostro esame finale in tempo reale durante la missione e a quanto ne so è stato anche l'unico programmatore di AGC che sia stato ritratto in un film (esattamente l'episodio Apollo 14 di 'Dalla Terra Alla Luna'). Lo stesso Don ne ha scritto di suo pugno (oltre a tante altre storie) in un modo piuttosto interessante, alquanto divulgativo.
Ma ecco la soluzione dettagliata, in parole che non devono differire di molto da quelle usate da Don in persona-
La procedura sviluppata per rimediare al segnale di fallimento della missione erroneo sull'Apollo 14 era questa:

- Dopo che cominciava il conto alla rovescia del NOUN 62 (un conto alla rovescia mostrato automaticamente sul display che si concludeva con l'accensione dei motori, detta iniezione) ma prima che i motori si accendessero effettivamente, settare il registro di modo a 71 ad indicare che un fallimento della missione era già accaduto. Osservate che i valori sono ottali, quindi 107 significa 71, bisognava battere sulla tastiera:

```
VERB 21 NOUN 1 ENTER 1010 ENTER
107 ENTER
```

- Esattamente 26 secondi dopo l'iniezione incrementare manualmente la manopola di accelerazione dei motori che spingevano la discesa fino al 100% .

- Per abilitare le equazioni di allunaggio. Le equazioni non si sarebbero abilitate automaticamente perché il registro di modo lo abbiamo appena settato al valore fittizio di 71. , digitare

```
VERB 25 NOUN 7 ENTER 101 ENTER
200 ENTER 1 ENTER
```

- Per disabilitare esplicitamente il controllore del fallimento missione. Ciò resettava il bit FLAGWORD che controllava il controllore stesso, digitare:

```
VERB 25 NOUN 7 ENTER 105 ENTER
400 ENTER 0 ENTER
```

- Per settare il registro di modo al valore giusto, che era 63, digitare

```
VERB 21 NOUN 1 ENTER 1010 ENTER
77 ENTER
```

- Riportare la manopola al suo valore minimo. La manopola sarebbe rimasta al 100% perché adesso era guidata dalle equazioni di allunaggio. Ma se si fosse omesso questo passaggio, la manopola sarebbe rimasta al 100% anche quando successivamente le equazioni avrebbero richiesto spinta parziale, con effetti non desiderabili.

Anche questa è programmazione.
Naturalmente, scommetto che adesso sembra ovvio, vero?

L'ABC di jQuery

I tuoi primi passi nell'uso del web-framework JavaScript più cool del momento

David Folletto Casali

Breve excursus storico: più o meno in modo coincidente all'onda Ajax sono nate alcune librerie javascript di automazione per lo sviluppo web. Lo scopo era rendere a tutti la vita più semplice. Nacque così la prima libreria famosa di questo tipo, chiamata *Prototype*. Questa però non è tutt'oggi molto più che una libreria di classi e funzioni: una serie di strumenti utili per semplificarsi la vita ma non molto oltre questo.

Inizialmente la documentazione era anche piuttosto scarsa e quindi l'adozione di Prototype per quanto crescente era però in qualche modo limitata. Per questo e altri motivi, qualche tempo dopo ha visto la luce *jQuery*, che si pone quasi in contrasto con Prototype. Vuole infatti porre non una serie di strumenti ma principalmente un modello uniforme estensibile e ricorsivo e su questo costruire un framework.

Con il trascorrere dei mesi Prototype si è evoluto, ma così anche jQuery e anche una serie di altre librerie sono diventate mature e interessanti: Dojo, YUI, MooTools (italiano!), etc.

Non voglio entrare nel merito di quale strumento sia meglio: ogni persona ha le sue preferenze e ogni progetto ha uno strumento ideale. Lasciando quindi alle preferenze di ciascuno approfondire o usare una specifica libreria, qui procedo a spiegare jQuery, che è forse quella dotata della struttura più atipica e che però personalmente trovo molto più "vicina" a quello che è lo stile di programmazione di JavaScript per il DOM.

Questa spiegazione farà anche accenno ad alcune funzionalità avanzate di JavaScript, cosa che può sempre tornare utile.

Architettura

Ma cosa è quindi jQuery? jQuery è uno strumento per rendere la manipolazione delle pagine (X)HTML uniforme e immediato, con il minor numero possibile di problemi fra un browser e l'altro.

La libreria jQuery si basa su alcuni punti cardine:

1. Il risultato di una chiamata jQuery è un altro oggetto jQuery.
2. Gli oggetti jQuery sono insiemi di elementi.
3. La chiave di tutto è la funzione `$()`.
4. Separazione della logica funzionale dalla logica strutturale (interazione vs presentazione).

Esempio

Ma allora, come si usa esattamente? Il suo funzionamento è quasi magico. Partiamo dall'esempio sulla homepage della libreria:

```
$("p.surprise").addClass("ohmy").show("slow");
```

E scomponiamolo:

- `$()` utilizza un selettore CSS (non c'è niente di nuovo da imparare quindi) e restituisce una collezione di oggetti jQuery che rappresentano tutti i tag che il selettore ha identificato. Può essere uno solo, o tutti quelli sulla pagina.
- A tutti i tag viene quindi aggiunta la classe "ohmy".
- A tutti i tag viene applicata la funzione *`show()`*, ovvero vengono visualizzati.

Si vede in modo chiaro come sia contemporaneamente semplice e potente. La combinazione della funzione `$()` con la possibilità di concatenare operazioni raggiunge una notevole potenza espressiva con pochissimo codice.

jQuery: poco codice con notevole potenza espressiva

Analisi

Vediamo come è fatto internamente. Faccio notare che non riproporrò praticamente mai il codice completo, che è piuttosto complesso, quanto alcuni estratti utili ai fini logici e di comprensione di javascript.

```
var jQuery = function(a, c) { ... }
var $ = jQuery;
```

La costruzione è semplice, anche se implica forse alcuni passaggi caratteristici di javascript:

1. Viene creato un oggetto jQuery (`var jQuery`) e gli viene assegnato un oggetto funzione (*function()* {}) tramite una lambda function.
2. Viene creato un oggetto $ al quale viene assegnato l'oggetto jQuery. jQuery era una funzione, quindi anche $ sarà una funzione.

Notate come in

```
var $ = jQuery;
```

jQuery sia passato senza le tonde finali delle funzioni. Scrivere "jQuery" significa utilizzare l'oggetto funzione passato per riferimento. Scrivere "*jQuery()*" significa chiamare la funzione associata all'oggeto jQuery.

jQuery 1: pragma

Dopo aver accennato ai principi fondamentali di jQuery (tutto nasce da $() e il risultato di un metodo jQuery è sempre un oggetto jQuery) vediamo un po' concretamente con qualche esempio come si può usare jQuery stesso.
Riprendo dall'esempio fatto in precedenza:

```
$("p.surprise").addClass("ohmy").show("slow");
```

Questa chiamata:

1. Seleziona tutti gli elementi p.surprise (selettore stile CSS).
2. Aggiunge a tutti gli elementi p.surprise la classe "ohmy" (class="ohmy").
3. Visualizza tutti gli elementi p.surprise (modificati con class="ohmy") con una animazione lenta.

Avete provato a pensare quante righe di codice avrebbe richiesto una modifica analoga, anche premettendo che avevate già la funzione che usa i selettori CSS?
Avete notato che tutto questo con jQuery si fa in una riga, in modo chiaro ed efficiente?
Proseguendo in modo intuitivo:

1. se c'è un *addClass()* c'è anche un *removeClass()*.
2. se c'è *show()* c'è anche *hide()*.

Così è. Non solo, dato che *show()* e *hide()* sono due operazioni distinte che spesso si accompagnano, la sintesi è la funzione *toggle()*, che non fa altro che chiudere quando è aperto e aprire quando è chiuso. Ovvero, *toggle()* è un interruttore.
Piccola variazione: avete visto il menu a scomparsa che io uso sul mio blog im.digitalhymn.com/ (si, la

barra nera insomma). E' evidente che si può realizzare in modo molto semplice con una piccola variazione del comando *toggle()*, che invece di aprirsi genericamente si apre regolando l'altezza. Questa operazione si chiama slide ed è utilizzabile con queste funzioni:

- *slideUp()* – chiude
- *slideDown()* – apre
- *slideToggle()* – interruttore

Piccola nota conclusiva a questo articolo. Le funzioni prendono tutte due parametri, entrambi opzionali:

1. `speed` – il primo parametro regola la velocità di apertura/chiusura
2. `callback` – il secondo parametro è una funzione che viene chiamata appena l'operazione si completa.

jQuery 2: Ajax

Come ogni buona libreria "Web 2.0" anche jQuery contiene il suo supporto Ajax. Dato l'orientamento di jQuery verso il DOM (come abbiamo già detto, jQuery inizia con il selezionare tutti gli elementi su cui si applicherà) una delle funzionalità primarie che vengono intuitivamente in mente è il supporto a quello che viene chiamato Ahah. Ovvero, una variante Ajax che invece si chiama "Asynchronous HTML And HTTP".

> jQuery ha il suo buon supporto Ajax (anche se lo chiama Ahah)

Non sono molto d'accordo con questa definizione che ritengo superflua: Javascript è comunque utilizzato, HTML dovrebbe essere XHTML (quindi XML) e HTTP è una ovvietà.

Caricamento diretto via Ajax

Comunque, a parte questa digressione, questo tipo di funzionalità non fa altro che caricare tramite Ajax il contenuto di un elemento della pagina:
Partiamo da un esempio:

```
$("#list").load("form/sets?user=" + $("#email").attr("value"))
```

Il comando è semplice ed intuitivo, come è prevedibile da jQuery:

1. Seleziono gli elementi con ID "list". Essendo un ID è univoco.
2. Carico (*load*) al suo interno il risultato della pagina puntata dall'URL passato come parametro.
3. Il parametro è composto dal valore dell'elemento con id "email", che non è altro che il campo del form contenente l'e-mail dell'utente.

Con una singola riga di codice ho interrogato il server, ottenuto il risultato e modificato la pagina.
Per rendere l'operazione un po' più user-friendly poi ho preceduto la riga di cui sopra con:

```
$("#list").html("Asking Flickr\ldots{}");
```

Che immediatamente cambia il contenuto del campo in modo che l'utente capisca subito che sta caricando qualcosa.

Dettagli su *load()*

Load accetta oltre all'URL anche due parametri opzionali: parametri e funzione callback.

1. L'URL è un normale URL, non credo vi sia niente da aggiungere.
2. I parametri sono costituiti da un hash contenente coppie di valori da passare (i.e. {test: true}).
3. La funzione callback è un oggetto funzione javascript (anche lambda, ovvero dichiarato al momento) che viene eseguito appena si è ricevuta risposta dal server. Questa funzione prende un parametro, data, che è il contenuto della risposta Ad es.

```
function(data) { alert("Ok!\n" + data); }
```

Ad esempio quindi la chiamata di cui sopra poteva diventare:

```
$('#list').load(root + 'form/sets',
                {user: $('#email').attr('value')});
```

Che è anche più chiaro da leggersi, separando la posizione della risorsa/comando (URL) dai parametri variabili passati. Ajax: anche *post()* e *get()*
Però *load()* è un comando che carica direttamente nella pagina. Ajax in realtà consente manipolazioni più complesse che potrebbero non richiedere un inserimento 1:1 diretto del risultato sulla pagina.
Per questo motivo ci sono *post()* e *get()* che funzionano con gli stessi parametri, ma con la differenza che non si applicano a nessun elemento del DOM:

```
$.post(url, params, callback);

$.get(url, params, callback);
```

Come vedete la chiamata non specifica nessun elemento. La funzione callback anche qui ha un parametro che contiene l'oggetto XML.

```
$.post("test.cgi", function(data) { alert("Ok! " + data); })
```

Preciso un dettaglio importante che potrebbe sfuggire ai meno esperti: si dovrebbe utilizzare GET nel caso in cui non si modifica alcuna risorsa sul server, mentre POST quando si inviano dati che modiicano lo stato del server.

JSON, cross domain loading

Vi è anche un'alternativa ad Ajax, che utilizza un interessante workaround in grado fra le varie cose di aggirare i problemi di sicurezza connessi con la chiamata Ajax allo stesso dominio di appartenenza. Per quanto possa portare a possibili problemi di sicurezza, tutt'ora non mi risulta ve ne siano, né che i browser lo impediscano in alcun modo.
JSON è un formato di interscambio di dati semplice, basato su sintassi JavaScript (JavaScript Object Notation). Di fatto si tratta di una serie di strutture dati javascript (hash, array, . . .).
La nostra libreria jQuery supporta anche questo tipo di chiamate:

```
$.getJSON(url, params, callback);
```

Qui bisogna capire un po' meglio come passare i dati, in quanto devono essere formattati non in XML ma in JSON, ma per il resto dei dati JSON sono immediatamente disponibili e non necessitano di parsing.

jQuery supporta JSON nativamente

Il parametro passato alla funzione callback infatti è l'oggetto JSON appena caricato.

Conclusioni

jQuery è un framework JavaScript per astrarre la programmazione lato client del comportamento di ogni singola pagina (X)HTML.
Pubblicato da John Resig, ha in realtà la prima versione stabile soltanto il 26 agosto del 2006.
Come si è visto è possibile in poche operazioni effettuare anche azioni complesse.
È presente anche una comoda gestione degli eventi, assieme alla loro propagazione; e non manca una sezione per l'utilizzo di AJAX, con utili e veloci funzioni per istanziare gli oggetti per effettuare la connessione / invio di dati.

Riferimenti

1. Il sito di John Resig ejohn.org/
2. Documentazione del framwork docs.jquery.com/
3. Libreria di plugin per jQuery per realizzare UI jqueryui.com/
4. Libreria di componenti JSF basati utilizzando i plugin di jQuery www.jquery4jsf.org/

Monitoraggio e interattività di ambienti chiusi tramite la Stereo Vision

Realizzazione di un sistema capace di interagire con il movimento nelle tre dimensioni di soggetti in ambiente chiuso monitorato da webcam in .NET

Francesco Bini, Giacomo Veneri, Marta Veneri, Luca Colucci

Lo scopo dell'articolo è descrivere un'applicazione 'low cost' che possa offrire prestazioni di buon livello nel campo dell'interazione, attraverso la multimedialità, con soggetti che popolano un ambiente chiuso (stanza, museo ecc).

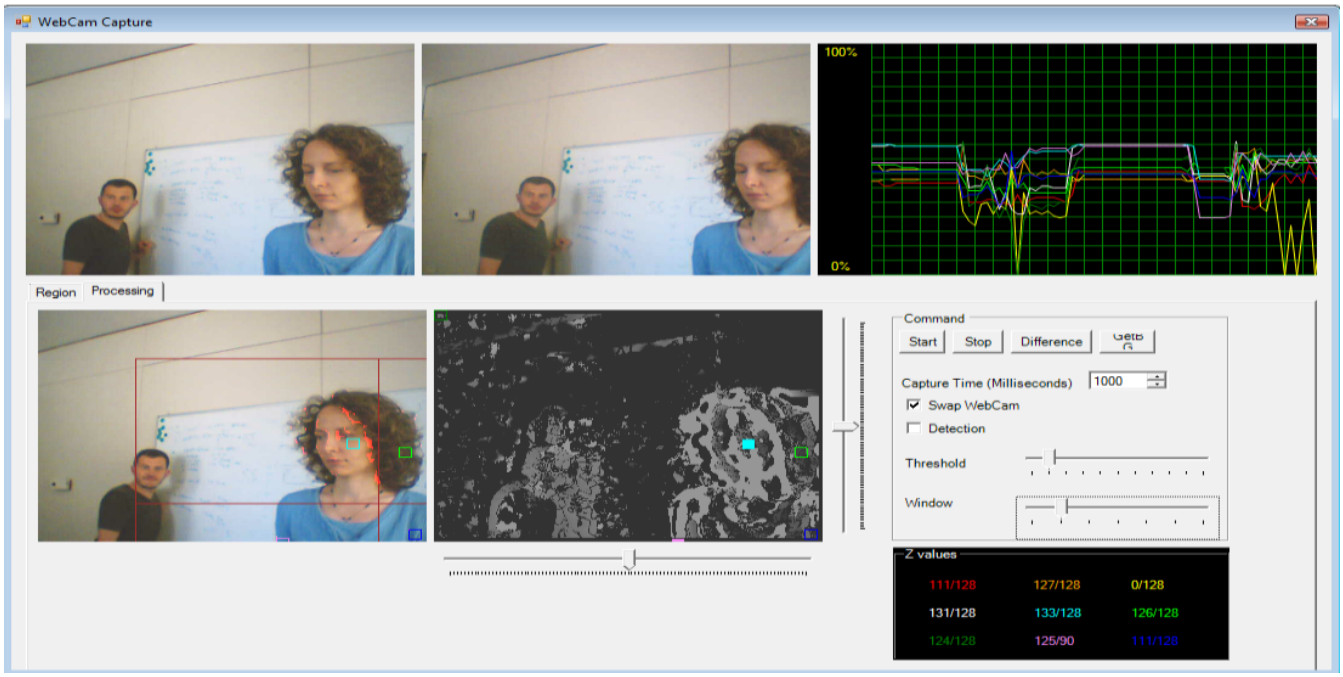
Per capire meglio il contesto si ricorre ad un esempio: la possibile applicazione all'interno di una sala di un museo. Il sistema è composto di due webcam collegate a un server che riprendono la scena da punti leggermente diversi ma simili e da altri computer dislocati all'interno della stanza, i quali ospitano i contenuti multimediali sulle opere esposte nei loro pressi (ad esempio illustrano la storia di un quadro e del suo autore). Si consideri l'immagine ripresa dalle telecamere divisa in settori di interesse (il centro, l'angolo in alto a sinistra ecc) e si supponga che un visitatore, nel corso del suo tour museale, si soffermi in una zona prossima ad un quadro. Il server centrale rileva la situazione (relativa alla posizione del soggetto) e dà l'input di riprodurre il filmato adeguato al PC nelle immediate vicinanze dell'opera: il visitatore può così vedere ed ascoltare spiegazioni aggiuntive sul dipinto osservato. L'articolo si propone di illustrare la realizzazione di un'applicazione simile sfruttando due semplici webcam e alcuni computer messi in rete.

L'idea su cui si basa il sistema proposto è quella di sfruttare principi di Stereo Vision contemporaneamente ad altri di Motion Detection.

Stereo Vision + Motion Detection

Ai primi sono legati algoritmi in grado di stimare, da due o più immagini contemporanee ma riprese da punti diversi, la profondità di una scena; i secondi sono utili per localizzare il movimento dei soggetti nella stanza osservata. Proseguendo con l'esempio del museo, si prova a illustrarne un caso pratico; naturalmente il sistema conosce preventivamente la disposizione dell'ambiente che deve monitorare. Esso ha preconfigurati i settori 'caldi' della scena, in altre parole sa esattamente dove (nelle 3 dimensioni x,y,z) sono esposte le opere di particolare interesse all'interno della stanza, quale PC le è dedicato ecc. Quando la struttura capta il movimento, riesce anche a dedurre in quale settore sta avvenendo; se la profondità (stimata) del soggetto che si sta muovendo è simile alla profondità (conosciuta) dell'opera collocata in quel settore significa che il visitatore è nelle vicinanze del dipinto; viene lanciato un segnale di input al PC relativo, con il comando di avviare la riproduzione del relativo documento multimediale descrittivo (vedi **Figura 1**).

Figura 1 — screenshot dell'applicazione principale



Stereo vision

La Stereo Vision (visione stereo) è una tecnica per dedurre la posizione 3D degli oggetti da due o più viste contemporanee di una scena. L'uomo ha due occhi complanari posti nella parte anteriore della testa; grazie a questa particolare disposizione ravvicinata e allineata, ogni occhio ha una visione della stessa scena da un angolo leggermente diverso. I due punti di vista degli occhi sono molto simili, ma ciascuno raccoglie informazioni visive che l'altro non può raccogliere (disparità fra le scene).

Ogni occhio ha la propria visuale e le due distinte immagini sono inviate al cervello per l'elaborazione: la mente umana è capace di utilizzare queste disparità per stimare la profondità della scena. La stima della disparità (in maniera rapida) è un problema complicato; le differenze possono essere causate da occlusioni di oggetti, riflessioni speculari, rumore del sensore e varie altre cause.

La Stereo Vision applicata ai computer si basa su due (o più) telecamere, separate da una piccola distanza che riprendono la stessa scena, simulando così il sistema visivo umano. Molti modelli computazionali di visione stereo, nel matching delle corrispondenze, assumono un vincolo di unicità: ogni caratteristica individuata in un'immagine deve essere trovato in una e una sola caratteristica nelle altre immagini [1].

> La Stereo Vision è una tecnica per dedurre la posizione 3D degli oggetti da due o più viste contemporanee di una scena.

Tale vincolo sembra essere plausibile, poiché negarlo, sarebbe equivalente a supporre che le caratteristiche della scena corrispondenti ai matching siano in due posti contemporaneamente. Il valore del vinco-

lo di unicità per eliminare false corrispondenze è stato dimostrato empiricamente in molti algoritmi stereo.

Ci sono numerosi metodi per calcolare la mappa delle disparità (disparity map), entrare troppo nel dettaglio esula dall'obiettivo dell'articolo, comunque il risultato è una nuova immagine in cui si nota una sorta di mappa 3D della scena osservata.

La **Figura 2** dimostra quanto detto in precedenza, in alto sono posizionate le due riprese di partenza della scena (telecamera di sinistra e di destra); l'immagine grande invece rappresenta la mappa di disparità delle due. Le zone più vicine in questo caso sono colorate con tinte chiare e, allontanandosi dal punto di ripresa delle telecamere, gli oggetti (o meglio la loro ricostruzione) diventano rappresentati con tonalità sempre più scure.

Motion Detection

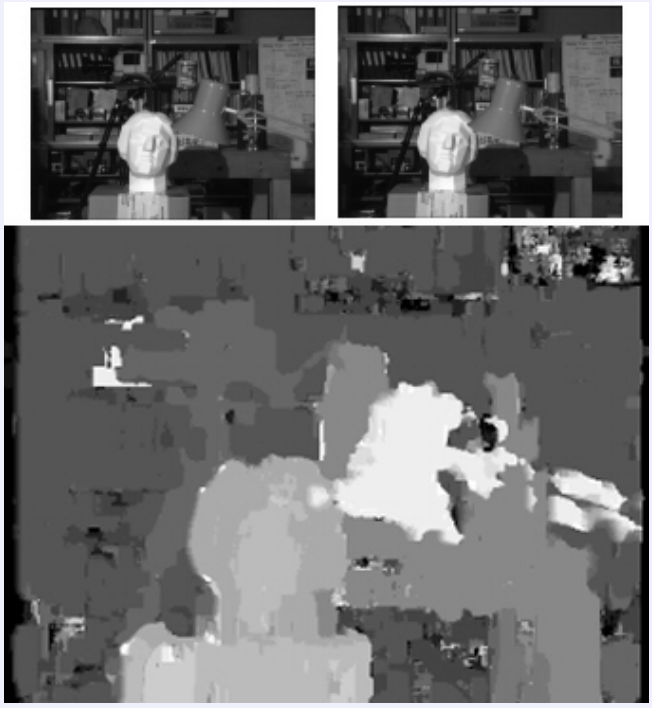
La Motion Detection è una tecnica utilizzata in vari settori (nell'industria per la localizzazione di oggetti in movimento, nei sistemi di videosorveglianza intelligente, negli applicativi software per editing dei video e nei videogiochi) ed è sostanzialmente un processo in grado di localizzare un oggetto in movimento nel tempo attraverso l'impiego di una telecamera. Dato un flusso video, è possibile, attraverso particolari algoritmi di analisi dei frames, localizzare e seguire uno o più oggetti che si muovono in una determinata area e quindi ricavare informazioni sulla posizione passata, su quella attuale e fare previsioni sullo stato futuro.

> Il Motion Detection è un processo in grado di localizzare un oggetto in movimento nel tempo attraverso l'impiego di una telecamera.

In generale, un sistema di Motion Detection è composto da due parti: la prima provvede a localizzare l'oggetto da tracciare, la seconda ne esegue il tracciamento effettivo.

Esistono vari approcci e uno dei più comuni e semplici consiste nel confrontare una copia in bianco e nero dell'attuale frame con una copia in bianco e nero di quello immediatamente precedente. Tale confronto avviene tramite l'impiego di tre filtri: il primo calcola la differenza tra il frame corrente e quello precedente, il secondo calcola la soglia di differenza tra i due frames (ovvero stima quanto sono differenti), il terzo elimina il rumore prodotto dalla telecamera. Tale approccio non è dei più precisi, ne esistono altri, infatti, che producono una rilevazione più dettagliata dell'oggetto in movimento, ma è utile nei casi in cui è sufficiente una stima dei cambiamenti dell'ambiente in esame.

Figura 2 — disparity map



Architettura del sistema

Il sistema è basato su di un'architettura server-client che sfrutta come canale di comunicazione il protocollo TCP. Il server centrale (SC), al quale sono connesse

due webcam che riprendono la scena (stanza), ospita il cuore dell'applicativo: si tratta di una windows form application in C# capace di utilizzare algoritmi di Stereo Vision e di Motion Tracking per capire se in una particolare regione della scena osservata ci sia del movimento, ed anche a quale profondità stia avvenendo questo movimento e, di reagire di conseguenza invocando, ad esempio, la riproduzione di un particolare audio o video. La scena ripresa dalle telecamere è suddivisa in una griglia configurabile di 9 settori (3 righe, 3 colonne); ogni client periferico (CP) è associato ad un settore tramite un parametro di configurazione. Il CP è una windows form application in C# capace di gestire un file multimediale ed espone dei metodi, invocabili da remoto tramite canale TCP, per avviare, fermare o mettere in pausa i files multimediali.

L'applicazione del SC è associata ad un'istanza della classe `FormStereo`, mentre i CP sono applicazioni associate ad istanze della classe `FormMedia`. Esiste anche un'interfaccia per la gestione degli oggetti remoti che è la `RemoteApplicationInterface` la quale espone dei metodi semplici: *start()*, *stop()*, *register()* ecc...

La classe `ApplicationServer` serve per aprire una connessione remota sul canale TCP a un particolare indirizzo URI ed esporre di conseguenza i metodi dell'interfaccia `RemoteApplicationInterface`; essa è chiamata all'avvio sia dai CP, sia dal SC attraverso il corrispondente metodo *StartServer()*. L'unica differenza sta nella classe di implementazione dell'interfaccia remota che hanno i due dispositivi: `StereoRemoteApplication` per l'SC e `MediaPlayerRemoteApplication` per i CP. Il CP, al proprio avvio, invoca anche il metodo *StartClient()* per poter dire al SC quale settore andrà a gestire e con quale URI sarà possibile invocarlo; per fare questo viene chiamato il metodo *Register()* dell'oggetto remoto appena ottenuto (il form del SC), che in questo caso particolare è proprio un'istanza della classe `StereoRemoteApplication`. Tale metodo non fa altro che invocare il corrispondente metodo *BuildClient()* della classe `MyCentroid` che, a sua volta, prepara la connessione TCP con il CP di turno rendendo di fatto possibile l'invocazione a runtime da parte del SC dei metodi dell'oggetto remoto di tipo `MediaPlayerRemoteApplication` (contenuto nel CP).

In ultima analisi si può affermare che la struttura server-client è bidirezionale: all'avvio i CP si registrano al SC per comunicargli quale settore gestiranno; a regime invece è il SC che funziona come un client invocando, secondo la sua logica di controllo, i metodi di gestione dei files multimediali dei vari CP.

Funzionamento del sistema

All'interno dell'applicazione principale, la windows form application in C# che sta sul SC, c'è bisogno di acquisire lo streaming video dalle webcam; per farlo si è scelto di adoperare un libreria in C#, chiamata DirectX.Capture [2]. Questa libreria si basa principalmente su DirectShow [3], un insieme di componenti COM che sono resi accessibili in C# dalla libreria .NET Interop. L'utilizzo della classe Capture per l'acquisizione del video è molto semplice poiché il grosso del lavoro è delegato a DirectShow.

Come si osserva dal **Listato 2**, durante il caricamento del `FormStereo` vengono istanziati due oggetti di tipo Capture per l'acquisizione dei video delle due webcam (sinistra e destra); ad ogni istanza viene associato un componente di tipo pictureBox che consentirà di vedere a schermo l'anteprima dei due streaming video. In seguito si inizializzano i 9 settori con la chiamata del metodo *adjustRegionRectangle()*; il principio con cui creare la griglia di 3 righe e 3 colonne è molto semplice: definendo posizione e dimensioni di un rettangolo centrale (interno al rettangolo di visualizzazione dei video) si ottiene una suddivisione dello schermo nei 9 settori voluti semplicemente prolungando i lati del rettangolo centrale fino ad intersecare il rettangolo esterno di visualizzazione. A ogni settore è associata un'istanza della classe `MyCentroid`.

Premendo il pulsante Start si attiva un Timer che ciclicamente invoca sempre lo stesso metodo *doDifference()*; questo è responsabile sia del calcolo della disparity map per la stima della profondità, sia del Motion Detection per il tracciamento dei movimenti.

Un Timer ciclicamente invoca un metodo che è responsabile sia del calcolo della disparity map per la stima della profondità, sia del motion detection per il tracciamento dei movimenti.

Per prima cosa si va ad acquisire una bitmap all'istante attuale invocando il metodo *CaptureControl()*; questo metodo consente di estrarre una bitmap da un controllo (il `pictureBox` in questo caso) sfruttando i metodi di due librerie di Windows per la gestione di interfacce grafiche: gdi32.dll e user32.dll. Le due immagini così ottenute servono per calcolare la disparity map con il metodo *doStereo()* della classe `Estereo`; la prima bitmap viene anche utilizzata per il Motion Detection, attraverso il metodo *ProcessFrame()* della classe `MyMotionDetector`. L'ultima parte del metodo *doDifference()* serve a controllare i settori; di ognuno viene calcolato ed aggiornato il centroide attraverso il corrispondente oggetto `MyCentroid`. Ogni istanza di tale classe ha una componente x ed y che sta ad indicare la relativa posizione spaziale quando nel settore di riferimento c'è del movimento (0,0 altrimenti), mentre la coordinata z rappresenta il valore medio (fra 0-nero e 255-bianco) dei pixel di quel settore. Ogni oggetto `MyCentroid`, e dunque ogni settore, ha configurato anche una soglia per il valore di z; il metodo *CheckProcess()* va appunto a vedere se in un particolare settore c'è del movimento contemporaneamente ad un valore z superiore alla soglia. Quando questa condizione è soddisfatta l'istanza della classe `MyCentroid` chiama il metodo *Start()* del corrispondente oggetto `RemoteApplicationInterface` (in pratica fa partire il file multimediale sul CP). Il SC invece si preoccupa solo dell'effetto visivo sullo schermo, cioè di visualizzare il centroide dei settori dove percepisce il movimento come dei rettangoli colorati totalmente, o solo nei bordi, a seconda della risposta del *CheckProcess()* del relativo oggetto `MyCentroid`.

Per il Motion Detection si è preso spunto da una libreria in C# [4] e, basandosi sull'interfaccia `IMotionDetector`, si è creata una nostra particolare istanza chiamata `MyMotionDetector`. Il principio su cui si basa quest'approccio è di confrontare l'attuale immagine (frame) con quella immediatamente precedente. Facendo riferimento al **Listato 4**, per capire quando c'è movimento, l'idea è di sottrarre due frame temporalmente consecutivi per vedere se esistono dei pixel differenti. Per rendere ancora più efficiente il procedimento, la differenza si fa fra copie dei frames in tonalità di grigio e si utilizzano anche alcuni filtri per eliminare il rumore. Calcolo il centroide dell'immagine risultante dalla sottrazione con il metodo *CalculateCentroid()* per capire in quale zona (x,y) è avvenuto il movimento ed in caso di movimento disegno anche nell'immagine originale il rettangolo del settore corrispondente. Infine si applica una serie di filtri all'immagine di partenza per vedere graficamente nell'applicazione l'effetto del movimento: i pixel che fanno parte dell'immagine differenza, e che, segnalano quindi la presenza di movimento, sono disegnati a schermo con il colore rosso.

Quando in un particolare settore c'è del movimento contemporaneamente a un valore di profondità superiore alla soglia configurata, si fa partire il file multimediale corrispondente.

Per la parte di stereo vision si è utilizzata `EStereo` [5], una libreria in C++ per la stima in tempo reale della disparity map fra due immagini.

Si è dovuto quindi procedere alla definizione di un oggetto che funzionasse come interfaccia fra C++ e C#: la classe `MyEstereo`, la quale delega tutto il lavoro di calcolo della disparity map alla classe in C++ `StereoMatching`, mentre nella propria implementa-

zione si occupa soltanto di modificare i parametri in input nel formato giusto per il metodo *doStereo()* della classe `StereoMatching` (da `Image` a `char*`). Spiegare nel dettaglio quest'ultimo metodo esula dallo scopo dell'articolo e pertanto basti sapere, al fine di poter seguire logicamente i passi del **Listato 5**, che il primo parametro di input (passato per riferimento) conterrà, dopo l'esecuzione del metodo, la bitmap di disparità.

Conclusioni

Quando si ha a che fare con ambienti che automaticamente reagiscono alle azioni dei soggetti che li popolano, si è portati a dare per scontato, che il cuore del sistema, in altre parole chi è deputato al controllo della logica d'interattività, sia il prodotto di chissà quale marchingegno o apparecchiatura hardware sofisticatissima. Ovviamente quando si parla di hardware avanzato tecnologicamente, va da sé che i costi totali dell'apparato vadano a lievitare in maniera esponenziale. L'articolo vuole semplicemente illustrare un metodo 'low cost' per ottenere dei risultati invidiabili nel campo dell'interazione con soggetti presenti in un ambiente chiuso con l'utilizzo di due semplici webcam, in altre parole con apparecchiature alla portata di qualsiasi tasca.

Possibili sviluppi futuri dell'applicazione potrebbero affinare il meccanismo con cui scatta l'allarme (rilevazione di movimento e stima della profondità). Difatti, poiché soprattutto la stima della profondità è un processo soggetto a forti disturbi rumorosi, si potrebbe creare un metodo che sia più robusto e meno soggetto a tali problematiche. Una possibile contromisura potrebbe essere quella di tenere in conto anche il fattore tempo: l'input di avvio dei CP (riproduzione del file multimediale) scatta se e solo se si supera la soglia sulla profondità per un certo numero di frame successivi (e non soltanto su quello corrente); si evitano così i casi in cui il limite sia oltrepassato solo per presenza stocastica di rumore .

Si potrebbe anche applicare un filtro per togliere il rumore di acquisizione delle webcam dalle immagini (il caratteristico effetto sale-pepe) prima di mandarle in input alla funzione che calcola la disparity map; basta trovare il giusto compromesso fra complessità di calcolo nell'applicazione del filtro e miglioramenti in termini di rumore. In altre parole, occorre che sia preservata una certa snellezza di calcolo in grado di consentire al sistema di reagire in real-time ad eventuali cambiamenti della scena senza appesantirsi troppo con calcoli che non lo farebbero trovare pronto al frame successivo da analizzare.

Riferimenti

1. Marr, D. & Poggio, T. *Science* 194, 283–287 (1976)
2. Brian Low, DirectX.Capture class library www.codeproject.com/KB/directx/directxcapture.aspx
3. Wikipedia l'enciclopedia libera., DirectShow it.wikipedia.org/wiki/DirectShow
4. Andrew Kirillov, Motion Detection Algorithms www.codeproject.com/KB/audio-video/Motion_Detection.aspx
5. `EStereo`, computer vision C++ library for real-time disparity estimation sourceforge.net/projects/estereo/

Listato 1 (parte 1 di 2): la struttura server-client

```
public class ApplicationServer {
    public ApplicationServer(String name, int port, Type commonInterfaceType) {
        TcpChannel tcpChannel = new TcpChannel(port);
        ChannelServices.RegisterChannel(tcpChannel, false);
        RemotingConfiguration.RegisterWellKnownServiceType(commonInterfaceType, name,
                                                           WellKnownObjectMode.SingleCall);
    }
}
public enum Status{START, STOP, PAUSE}
public interface RemoteApplicationInterface {
  void Start();
  void Stop();
  void Pause();
  Status GetStatus();
  void Register(string uri,int index);
  void Unregister(string uri,int index);
}
public class StereoRemoteApplication : MarshalByRefObject,Remote.RemoteApplicationInterface {
    public void Register(string uri, int index) {
    FormStereo.NewInstance().Register(uri,index);
  }
  .............
}
public class FormStereo : System.Windows.Forms.Form {
    private static FormStereo form;
    private MyCentroid[] regions = new MyCentroid[9];
    public static FormStereo NewInstance(){
     if (form == null){form = new FormStereo();}
    return form;
  }
  private FormStereo() {
    .....initialize components......
    StartServer();
  }
  private void StartServer() {
    ApplicationServer server = new ApplicationServer(ConfigurationSettings.AppSettings["NAME"],
                                                     ConfigurationSettings.AppSettings["PORT"],
                                                     typeof(StereoRemoteApplication));
  }
  public void Register(String uri, int index) {
    this.regions[index-1].BuildClient(uri);
  }
  ...............
}
public class MyCentroid {
    private String uri;
    private RemoteApplicationInterface remoteObject;
    public bool BuildClient(String uri) {
        Type requiredType = typeof(RemoteApplicationInterface);
        this.remoteObject = (RemoteApplicationInterface)Activator.GetObject(requiredType,uri);
        return true;
    }
    ............
}
public class MediaPlayerRemoteApplication : MarshalByRefObject,Remote.RemoteApplicationInterface {
  public void Start(){FormMedia.NewInstance().Start();}
  public void Stop(){FormMedia.NewInstance().Stop();}
  public void Pause(){FormMedia.NewInstance().Pause();}
  public Remote.Status GetStatus(){return FormMedia.NewInstance().GetStatus();}
  public void Register(string uri, int index){throw new NotImplementedException();}
  public void Unregister(string uri, int index){throw new NotImplementedException();}
}
[...]
```

Listato 1 (parte 2 di 2): la struttura server-client

```
public partial class FormMedia : Form,{
  private static FormMedia form;
  private AxWMPLib.AxWindowsMediaPlayer Player;
  public static Form1 NewInstance(){ if (form == null){form = new FormMedia();} return form;  }
  private FormMedia() {  .....initialize components......
    StartServer();
    StartClient();
  ... }
  private void StartServer() {
    ApplicationServer server = new ApplicationServer(MyName, MyPort,
                                    typeof(MediaPlayerRemoteApplication)); }
  private void StartClient() {
    Type requiredType = typeof(RemoteApplicationInterface);
    RemoteApplicationInterface remoteObject = (RemoteApplicationInterface)Activator.GetObject(
                                              requiredType, ConfigurationSettings.AppSettings["SERVER"]);
    remoteObject.Register(ConfigurationSettings.AppSettings["URI"],
                                  ConfigurationSettings.AppSettings["INDEX"]);  }
  public void Start() { this.Player.Ctlcontrols.play(); }
  public void Stop() { this.Player.Ctlcontrols.stop(); }
  public void Pause() { this.Player.Ctlcontrols.pause(); }
  public Status GetStatus() {
    if(this.Player.playState==WMPLib.WMPPlayState.wmppsPlaying) return Status.START;
    if(this.Player.playState==WMPLib.WMPPlayState.wmppsStopped) return Status.STOP;
    if(this.Player.playState==WMPLib.WMPPlayState.wmppsPaused) return Status.PAUSE;
    return Status.STOP;
} }
```

Listato 2: Caricamento form

```
public class FormStereo : System.Windows.Forms.Form { ...
  private Capture capture1 = null;
  private Capture capture2 = null;
  private Filters filters = new Filters();
  private MyCentroid[] regions = new MyCentroid[9];
  private NumericUpDown rectangleX, rectangleY, rectangleWidth, rectangleHeight;
  private PictureBox pictureBox1,pictureBox2;
    .......
    private void FormStereo_Load(object sender, EventArgs e) { //acquisizione video e anteprima sui picturebox
      capture1 = new Capture(filters.VideoInputDevices[0], filters.AudioInputDevices[0]);
      capture2 = new Capture(filters.VideoInputDevices[1], filters.AudioInputDevices[0]);
      capture1.PreviewWindow = this.pictureBox1;
      capture2.PreviewWindow = this.pictureBox2;
      adjustRegionRectangle();    //inizializzazione dei 9 settori
      ......
    }
    private void adjustRegionRectangle() {
      ...controllo che le dimensioni del rettangolo centrale siano ammissibili....
        //creo le 9 regioni (3 righe e 3 colonne) e ad ognuna associo un oggetto MyCentroid
      int xUpLeft = 0; int yUpLeft = 0;          //prima riga
      int widthReg = (int)this.rectangleX.Value;
      int heightReg = (int)this.rectangleY.Value;
      regions[0] = new MyCentroid("REGION1",xUpLeft, yUpLeft, widthReg, heightReg, regionColors[0]);
      xUpLeft = (int)this.rectangleX.Value;  widthReg = (int)this.rectangleWidth.Value;
      regions[1] = new MyCentroid("REGION2", xUpLeft, yUpLeft, widthReg, heightReg, regionColors[1]);
      xUpLeft = (int)this.rectangleX.Value + (int)this.rectangleWidth.Value; widthReg = widthTot - xUpLeft;
      regions[2] = new MyCentroid("REGION3", xUpLeft, yUpLeft, widthReg, heightReg, regionColors[2]);
      ......
 } }
```

Listato 3 (parte 1 di 2): *doDifference()*

```
public class FormStereo : System.Windows.Forms.Form {
    private PictureBox pictureBox3,pictureBox4;
    Estereo eStereo = new Estereo();
    private MyMotionDetector detector = new MyMotionDetector();
    .......
    private void doDifference()
    {
      Bitmap bmp1 = CaptureControl(this.pictureBox1);
      //motion detector
      detector.ProcessFrame(ref bmp1);
      pictureBox4.Image = bmp1;
      Bitmap bmp2 = CaptureControl(this.pictureBox2);
      Bitmap stereo = null;
      //disparity map
      if (swapCheck.Checked)
      {
          stereo = (Bitmap)eStereo.doStereo(bmp2, bmp1, this.window.Value, this.threshold.Value, 24,
           this.xOffset.Value, this.yOffset.Value);
      }
      else
      {
          stereo = (Bitmap)eStereo.doStereo(bmp1, bmp2, this.window.Value, this.threshold.Value, 24,
           this.xOffset.Value, this.yOffset.Value);
      }

      this.pictureBox3.Image = stereo;
      Graphics g = Graphics.FromImage(stereo);
      //ridisegno eventuale dei centroidi delle regioni (settori)
      foreach (MyCentroid region in regions)
      {
          if (region.Active)
          {
              region.Z = calculateMeanValue(stereo, region.Rect);
              if (region.Z > 0)
              {
                  bool b = region.CheckProcess();
                  if (b)
                  {
                      g.FillRectangle(new SolidBrush(region.Color), (float)region.X, (float)region.Y,
                                      10f, 10f);
                  }
                  else
                  {
                      g.DrawRectangle(new Pen(region.Color), (float)region.X, (float)region.Y,
                                      10f, 10f);
                  }
              }
          }
      }
      ......
    }
    .....
```

Listato 3 (parte 2 di 2): *doDifference()*

```
[DllImportAttribute("GDI32.DLL")]
  private static extern bool BitBlt(  IntPtr hdcDest,  int nXDest,  int nYDest,  int nWidth,
                                int nHeight,  IntPtr hdcSrc,  int nXSrc,  int nYSrc,  int dwRop);

  private const int WM_PRINT = 0x317, PRF_CLIENT = 4,PRF_CHILDREN = 0x10, PRF_NON_CLIENT = 2,
                    COMBINED_PRINTFLAGS = PRF_CLIENT | PRF_CHILDREN | PRF_NON_CLIENT;

  [DllImport("USER32.DLL")]
  private static extern int SendMessage(IntPtr hWnd, int Msg, IntPtr wParam,int lParam);

  public Bitmap CaptureControl(Control control)
  {
    Bitmap controlBmp;
    using (Graphics g1 = control.CreateGraphics())
    {
        controlBmp = new Bitmap(control.Width, control.Height, g1);
        using (Graphics g2 = Graphics.FromImage(controlBmp))
        {
            IntPtr dc1 = g1.GetHdc();
            IntPtr dc2 = g2.GetHdc();
            SendMessage(control.Handle, WM_PRINT, dc1, COMBINED_PRINTFLAGS);
            BitBlt(dc2, 0, 0, control.Width, control.Height, dc1, 0, 0, 13369376);
            g1.ReleaseHdc(dc1);
            g2.ReleaseHdc(dc2);
        }
    }
    return controlBmp;
  }
}

class MyCentroid
{
    private RemoteApplicationInterface remoteObject;
    protected double x, y, z, zTreshold = 0;
    ....
    public bool CheckProcess()
  {
    //guardo se c'Ã¨ movimento e se siamo sopra soglia
    if (((this.x > 0) || (this.y > 0)) && (this.z > this.zTreshold))
    {
      if ((this.remoteObject != null) && (this.remoteObject.GetStatus() != Status.START))
      {
          this.remoteObject.Start();
          return true;
      }
      return true;
    }
    return false;
  }
  ....
}
```

Listato 4 (parte 1 di 2): motion detector

```
class MyMotionDetector : IMotionDetector
{
    // Process new frame
    public void ProcessFrame( ref Bitmap image )
    {
        if ( backgroundFrame == null )
        {
            // create initial backgroung image
            backgroundFrame = grayscaleFilter.Apply( image );

            // get image dimension
            width   = image.Width;
            height  = image.Height;

            // just return for the first time
            return;
        }
          Graphics g = Graphics.FromImage(image);
        Bitmap tmpImage;

        // apply the grayscale file
        tmpImage = grayscaleFilter.Apply( image );

        // set backgroud frame as an overlay for difference filter
        differenceFilter.OverlayImage = backgroundFrame;

    // apply difference filter
    Bitmap tmpImage2 = differenceFilter.Apply( tmpImage );

    // lock the temporary image and apply some filters on the locked data
    bitmapData = tmpImage2.LockBits( new Rectangle( 0, 0, width, height ),
        ImageLockMode.ReadWrite, PixelFormat.Format8bppIndexed );

    // threshold filter
    thresholdFilter.ApplyInPlace( bitmapData );
    // erosion filter
    Bitmap tmpImage3 = erosionFilter.Apply( bitmapData );

    // unlock temporary image
    tmpImage2.UnlockBits( bitmapData );
    tmpImage2.Dispose( );

    // calculate amount of changed pixels
        pixelsChanged = ( calculateMotionLevel ) ?          CalculateWhitePixels( tmpImage3 ) : 0;

    foreach(MyCentroid region in this.regions){
       if (region.Active)
       {
           double[] centroid = CalculateCentroid(tmpImage3, region.Rect);
           region.X = centroid[0];
           region.Y = centroid[1];
           if ((region.X > 0) && (region.Y > 0))
           {
               g.DrawRectangle(new Pen(region.Color), (float)region.X, (float)region.Y, 10f, 10f);
               g.DrawRectangle(new Pen(Color.Brown), region.Rect);
           }
       }
    }
```

Listato 4 (parte 2 di 2): motion detector

```
[...]

        // dispose old background
        backgroundFrame.Dispose( );
        // set backgound to current
        backgroundFrame = tmpImage;

        // extract red channel from the original image
        Bitmap redChannel = extrachChannel.Apply( image );

        //  merge red channel with moving object
        mergeFilter.OverlayImage = tmpImage3;
        Bitmap tmpImage4 = mergeFilter.Apply( redChannel );
        redChannel.Dispose( );
        tmpImage3.Dispose( );

        // replace red channel in the original image
        replaceChannel.ChannelImage = tmpImage4;
        Bitmap tmpImage5 = replaceChannel.Apply( image );
        tmpImage4.Dispose( );

        image.Dispose( );
        image = tmpImage5;

    g.Dispose();
    }
}
```

Listato 5 (parte 1 di 2): stereo vision

```
....from MyEstereo.h.....
#pragma once
#using <mscorlib.dll>
#include <vcclr.h>
#using <system.drawing.dll>

public ref class MyEstereo
{
public:
    MyEstereo(void);
    System::Drawing::Image^ doStereo(System::Drawing::Image^ leftImage,
    System::Drawing::Image^ rightImage, int win, int threshold,
     int depth, int xOffset, int yOffset);
};

[...]
```

Listato 5 (parte 2 di 2): stereo vision

```
....from MyEstereo.cpp.....
........
System::Drawing::Image^ MyEstereo::doStereo(System::Drawing::Image^ leftImage,
 System::Drawing::Image^ rightImage, int win, int threshold,
  int depth, int xOffset, int yOffset) {
    StereoMatching* sm = new StereoMatching();
    sm->initializeContext();
    sm->setHoropter(0);
    sm->setLeftImageXOffset(xOffset);
    sm->setLeftImageYOffset(yOffset);
    sm->setAcceptDisparityThreshold(threshold);
    sm->setCorrelationWindowSize(win,win);
    sm->setInputImageSize(leftImage->Width,leftImage->Height);
    sm->setScale(0);
    sm->setNumDepth(depth);
    StereoImage* stereoImage = new StereoImage();
        stereoImage->alloc(leftImage->Width,leftImage->Height);
    unsigned char* leftBuffer = Image2Char(leftImage);
    unsigned char* rightBuffer = Image2Char(rightImage);
    sm->doStereo(stereoImage, leftBuffer, rightBuffer);
    delete sm;
    //todo reassign
    System::Drawing::Image^ img = Char2Image(stereoImage->imDepth8u, stereoImage->width,
                                                                        stereoImage->height);
    delete stereoImage;
    delete[leftImage->Width,leftImage->Height] leftBuffer;
    delete[rightImage->Width,rightImage->Height] rightBuffer;

    return img;
    //System::Diagnostics::Debug::WriteLine(" " + sizeof(stereoImage->valid_pixels));
};
unsigned char* Image2Char(const System::Drawing::Image^ image) {
    System::Drawing::Bitmap^ imageBmp =  gcnew System::Drawing::Bitmap((System::Drawing::Image^)image);
    unsigned char* res = new unsigned char[imageBmp->Height * imageBmp->Width];
    System::Drawing::Imaging::BitmapData^ data = imageBmp->LockBits(System::Drawing::Rectangle(0, 0,
                                                                imageBmp->Width, imageBmp->Height),
        System::Drawing::Imaging::ImageLockMode::ReadOnly,
           System::Drawing::Imaging::PixelFormat::Format32bppArgb);
    int offset = data->Stride - imageBmp->Width;
    int stride = data->Stride;
    UINT* ptr = (UINT*)data->Scan0.ToPointer();
    for (int y = 0; y < imageBmp->Height; y++)
    {
        for (int x = 0; x < imageBmp->Width; x++)
        {
            UINT* px = ptr+ y * stride / 4 + x;
            res[y*imageBmp->Width+x] = 0.3* (unsigned char)(*(px)>>16)
                                                        + 0.59*(unsigned char)(*(px)>>8)
                                                        + 0.11*(unsigned char)(*(px));
        }
    }
    imageBmp->UnlockBits(data);
    delete data;
    return res;
}

System::Drawing::Image^ Char2Image(unsigned char* buff, int w, int h) {
    System::Drawing::Bitmap^ imageBmp =  gcnew System::Drawing::Bitmap(w,h,
                                        System::Drawing::Imaging::PixelFormat::Format32bppArgb);
    for (int i=0; i<h; i++) {
        for (int j=0; j<w; j++) {
            int lum = 50+(buff[i*w+j]*4);
            imageBmp->SetPixel(j,i,System::Drawing::Color::FromArgb(lum,lum,lum));
        }
    }
    return imageBmp;
}
....
```

Traduzione statistica automatizzata

Come fanno i computer a tradurre i testi da una lingua ad un'altra?

Michael Nielsen

Come fanno i computer a tradurre i testi da una lingua ad un'altra? I traduttori umani, per tradurre i diversi significati, tutte le sfumature, che una parola o una frase possono assumere in contesti diversi, usano una robusta dose di conoscenza sul mondo.

Questo suggerisce che la traduzione automatizzata possa essere una sfida difficile, il genere di sfida che richiederebbe l'intelligenza artificiale.

Eppure, sebbene i sistemi di traduzione automatica, come per esempio *Google Translate* o *Yahoo!'s babel Fish* siano assolutamente perfettibili, riescono a fare un lavoro sorprendentememte buono. Come si spiega?

Introduzione alla traduzione statistca automatizzata

In quest'articolo, descrivo le idee che sono alla base dell'approccio di maggior successo alla traduzione automatizzata, fatta dalle macchine; approccio che va sotto il nome di *traduzione statistica automatizzata*.

In quest'approccio si parte da un insieme molto vasto di traduzioni di buona qualità, ovvero un cosiddetto *corpus* di testi (ad esempio i documenti delle Nazioni Unite) che sono stati già tradotti da umani in diverse lingue; quindi si impiegano quelle traduzioni già fatte per inferire un modello statistico di traduzione. Questo modello viene poi applicato a nuovi testi per fargli *indovinare* una traduzione ragionevole.

Non sono un esperto del settore. Ho scritto questi appunti per aiutarmi a internalizzare le idee fondamentali della disciplina. Tuttavia spero che questi appunti siano utili ad altri come panoramica introduttiva alla traduzione automatizzata statistica e come punto di partenza per ulteriori approfondimenti.

Formulare il problema

Immaginiamo di avere un testo in francese, f e che vogliamo trovare, di questo testo, una buona traduzione in inglese, e. Le traduzioni possibili di f in inglese, naturalmente, sono molte e traduttori diversi avranno opinioni diverse su quale sia la migliore. Possiamo modellare questa diversità di opinioni con una distribuzione di probabilità $P(e|f)$; questa è una distribuzione sull'insieme di traduzioni possibili .

$$e \text{ data } f$$

Un modo ragionevole di individuare la traduzione migliore è quello di scegliere la traduzione e che massimizza questa distribuzione di probabilità, ovvero la traduzione che sarà giudicata essere la migliore dal maggior numero di traduttori. Ripetiamo, si tratta di una legge di probabilità condizionata $P(e|f)$.

Il problema, con questa strategia, è che noi non sappiamo quale sia, questa funzione di probabilità, sappiamo solo che esiste.

Per risolvere questo problema, supponiamo di essere in possesso di un corpus di documenti che sono sia in francese che in inglese, ad esempio i documenti delle Nazioni Unite, oppure gli atti parlamentari canadesi. Useremo questo corpus per inferire un modello che *stima* la nostra probabilità condizionale $P(e|f)$.

Il modello che costruiremo non sarà perfetto ma con un corpus sufficientemente esteso e di qualità sufficientemente buona, otterremo delle traduzioni piuttosto soddisfacenti.

Per semplificare la trattazione, assumiamo che e ed f siano singole frasi e trascureremo anche la punteggiatura. Naturalmente, il traduttore che otterremo potrà essere applicato sequenzialmente a una serie di frasi.

Adesso, come si fa ad inferire dal corpus un modello che approssimi $P(e|f)$?

L'approccio standard è quello di usare il teorema di Bayes che ci consente di riformulare la nostra legge $P(e|f)$ come .

$$P(e|f) = \frac{P(f|e)P(e)}{P(f)}$$

Siccome f è data e $P(f)$ si trova al denumeratore, la massimizzazione di questa funzione è equivalente alla massimizzazione _sul solo_ numeratore $P(f|e)P(e)$.

Useremo, dunque, i dati di cui disponiamo per inferire modelli per $P(f|e)$ e per $P(e)$ e poi useremo quei modelli per cercare una e che massimizzi $P(f|e)P(e)$.

Questo modo di procedre può sembrare, *a priori*, discutibile. Perché non risparmiamo tempo e sforzo inferendo un modello per $P(e|f)$ e massimizziamo questo direttamente?

Ebbene risulta che la procedura indicata sopra, quella che impiega il teorema di Bayes, dia risultati migliori. Adesso, francamente, io non so esattamente la ragione per cui questo accade, secondo quanto riporta, in merito, la letteratura. Ma intuisco che quello che succede è questo: se lavoriamo direttamente su $P(e|f)$ accadrà che il nostro modello attribuirà probabilità elevate a frasi inglesi mal formate o scorrette. Il perché lo vedremo più avanti —risulterà ovvio dalla costruzione— ma per ora prendiamo per buona questa affermazione. Se lavorassimo direttamente su $P(e|f)$ le frasi malformate si vedrebbero riconosciuto lo status di traduzioni valide, e questo non è un buon risultato.

Si potrebbe osservare che la stessa cosa possa accadere con $P(f|e)$ e con $P(e)$. Ma bisogna considerare questi due fatti:

- i modelli per queste due formule saranno diversi e quindi la probabilità che *entrambi* attribuiscano alta probabilità *alla stessa* frase scorretta è molto piccola.
- Possiamo incorporare nel modello per $P(e)$ un controllo sintattico o grammaticale; quindi le frasi scorrette a cui venga attribuità una probabilità alta, saranno molto poche. Non lo faremo in questo articolo, ma è semplice da farsi.

Fin qui abbiamo suddiviso il problema della traduzione automatica in tre sottoproblemi:

- costruire un modello per la lingua che ci consenta di stimare $P(e)$
- costruire un modello della traduzione che ci consenta di stimare $P(f|e)$
- cercare la e che massimizza il prodotto $P(f|e)P(e)$

Ognuno di questi tre problemi è di per se ricco e può essere risolto in molti modi diversi. Nelle prossime tre sezioni vedremo metodi semplici per affrontare ognuno di essi.

Il modello della lingua

Supponiamo di suddividere una frase inglese e in una sequenza di parole $e = e_1 e_2 e3 \cdots e_m$.

Poi possiamo scrivere la probabilità di e (cioé la probabilità che occorra e) come il prodotto delle probabilità condizionali che occorrano le singole parole:

$$P(e) = \prod_{j=1}^{m} P(e_j|e_1 \cdots e_j - 1)$$

Così, ad esempio, per il frammento di frase "to be or not to..." (essere o non ...) la probabilità condizionale che l'ultima parola sia "be" ("essere") sarà *molto* più alta della probabilità che l'ultima parola sia una parola a caso presa dal testo (essere o non ... innaffiare).

La sfida che questo modello ci pone è quella di dover calcolare una quantità molto grande di probabilità condizionali distinte.

Per semplificare la questione introduciamo alcune assunzioni sulla forma che tali probabilità condizionali potranno assumere: la più drastica di tali assunzioni è quella per cui decidiamo che la probabilità che una parola occorra è *indipendente* dalla parola che la precede.

Quindi:

$$P(e_j|e_1 \cdots e_j - 1) = P(e_j)$$

e quindi

$$P(e) = \prod_{j=1}^{m} P(e_j|e_1 \cdots e_j - 1) = \prod_{j=1}^{m} P(e_j)$$

Potremmo stimare la probabilità che occorra una certa parola e_j prendendo un grosso corpus di testi in inglese e contando le parole.

Il problema di questo modello è che è irrealistico. Un modello più realistico è il *bigram model* nel quale la probabilità che occorra una certa parola dipende da quale parola occorra immediatamente prima, così:

$$P(e_j|e_1 e_2 \cdots e_m) = P(e_j|e_{j-1})$$

Ancora più realistico sarebbe il *trigram model* nel quale la probabilità che occorra una parola dipende dall'occorrenza delle *due* parole che precedono: $P(e_j|e_1e_2 \cdots e_m) = P(e_j|e_\{j-1\}e_\{j-2\})$.

Per rendere la trattazione che segue più realistica possibile, concentriamoci sul problema di come stimare le probabilità condizionali nel *bigram model*. Il *trigram model* si avvarrà naturalmente di osservazioni molto simili.

Il modo ovvio di procedere è di prendere un corpus molto esteso di testi inglesi e *contare* il numero di occorrenze di ogni singola coppia di parole e_1e_2 e poi stabilire che la probabilità che occorra la parola e_2, se l'ultima parola è e_1, è :

$$P(e_2|e_1) = \frac{\sharp(e_1e_2)}{\sharp e_1}$$

Il problema è che esiste una quantità *molto elevata* di coppie. Se assumiamo che la lingua inglese ha 50.000 parole, esistono due milioni e mezzo di coppie. Anche se si considera un corpus di testi esteso, ad esempio di un miliardo di parole, esiste una probabilità non banale che alcune coppie non compaiano nel testo e quindi ad esse venga assegnata una probabilità di occorrenza uguale a zero.

Mentre a priori non possiamo escludere che quelle coppie che non compaiono vengano considerate, durante la traduzione verso il francese.

Cioé, la procedura di addestramento probabilmente sottostimerà la probabilità (di occorrenza) di coppie che non compaiono nell'insieme di addestramento, mentre sovrastimerà la probabilità di coppie che compaiono.

Con i trigram è anche peggio.

Per questo problema non c'è una soluzione che sia ovviamente migliore delle altre. Sono state tentate molte soluzioni *ad hoc* e una mia ricerca veloce in letteratura suggerisce che non c'è neanche un largo consenso su quale sia l'approccio migliore. Giusto per dare un'idea di quali siano le opzioni in circolazione, riporto qui due proposte, adattate dalla ricerca di cui nella sezione 2.3 della tesi di dottorato di Philip Clarkson [1]

Il primo approccio è quello di abbandonare lo schema dei trigram puro e di adottare l'*interpolazione lineare* fra il modello monogram e quello bigram. E' probabile che un corpus di testo sufficientemente esteso e diversificato fornisca buone stime per quasi tutte le probabilità di una singola parola $P(e_1)$ quindi un modo per stimare la probabilità che occorra e_2 se è preceduta da e_1 (cioé $P(e_2|e_1)$) è:

$$P(e_2|e_1) = \lambda \frac{\sharp\, e_2}{N} \mid (1-\lambda)\, \frac{\sharp\, (e_1e_2)}{\sharp\, e_1}$$

dove N è il numero totale delle parole nel corpus. λ è un parametro che oscilla fra zero e uno ($0 < \lambda < 1$) e che deve essere determinato opportunamente. Il che può esser fatto usando un secondo corpus di testo e assegnando a λ un valore che massimizzi la probabilità media di ogni coppia in quel corpus.

Un secondo approccio è di usare un *fattore di riduzione* delle probabilità condizionali delle coppie che appaiono nel corpus di addestramento, sostanzialmente riducendo, così, la loro probabilità di occorrenza.

Il modo più facile è di moltiplicarle tutte per qualche costante c che varia fra zero e uno ($0 < c < 1$) e poi distribuire la probabilità che mimane, $1 - c$, uniformemente a tutte le coppie che non appaiono nel corpus.

Per concludere questa sezione, lasciatemi ricordare il fatto che il problema della creazione di un modello della lingue è un problema di grande interesse anche per coloro che si occupano di software di riconoscimento vocale. Infatti, molta della letteratura di ricerca sui modelli della lingua riguarda parimenti il riconoscimento del linguaggio parlato come la traduzione statistica.

La creazione di un modello della lingue è un problema di grande interesse anche per coloro che si occupano di software di riconoscimento vocale

Il modello della traduzione

> Nota: io parlo solo inglese, quindi non sono in grado di fornire esempi di traduzione. Tutti gli esempi che cito sono presi da Brown e altri (portal.acm.org/citation.cfm?id=92860).

In questa sezione costruiremo un semplice modello per la traduzione che ci consenta di calcolare $P(f|e)$

Intuitivamente, la traduzione di una frase consiste nel generare parole nella lingua di destinazione (possibilmente in un modo che tiene conto del contesto) a partire da parole della frase nella lingua d'origine.

Nella coppia di frasi

```
( Jean aime Marie | John loves Mary )
```

intuitivamente avvertiamo che a "Jean" corrisponde "John", a "Marie" corrisponde "Mary" e che a "aime" corrisponde "loves".

Naturalmente non è necessario che la corrispondenza fra le parole sia uno a uno e neanche che l'ordine in cui compaiono parole tradotte sia lo stesso ordine in cui comparivano le parole nella lingua originale.

A volte una parola in inglese può generare due o più parole francesi, a volte può non generarne nessuna.

Nonostante queste difficoltà, la nozione di una corrispondenza fra le parole è talmente utile che la formalizziamo con il concetto di allineamento.

Invece che dandone una definizione formale, fatemelo spiegare con alcuni esempi, cos'è un allineamento. Prendiamo la coppia di frasi

```
( Le chien est battu par Jean |
  John(6) does beat (3,4) the(1) dog(2) ).
```

In questo allineamento d'esempio la coppia di parentesi che contiene il 6 ci dice che "John" corrisponde alla sesta parola della frase in francese, cioé "Jean". La parola "does" non ha parentesi, il che ci indica che non corrisponde a nessuna parola nella frase in francese; "beat" corrisponde non ad una ma a due diverse parole nella frase in francese, la terza e la quarta, che sono "est" e "battu". E così via.

L'allineamento ci consente di individuare due nozioni particolarmente utili nella creazione di un modello per la traduzione: la prima è la *fertilità* ovvero il numero di parole francesi che corrispondono ad una sola parola inglese. Quindi nell'esempio precedente "does" ha fertilità 0, perché non le corrisponde nessuna parola francese. D'altra parte "beat" ha fertilità 2 perché le corrispondono due parole francesi.

La seconda nozione è la *distorsione* . In molti casi, le parole francesi che corrispondono a parole inglesi si trovano in una posizione simile nella frase cui appartengono, cioé magari più vicino all'inizio della frase, al centro oppure più vicine alla fine della frase. Daremo una definizione più formale di questa nozione a breve.

Costruiremo il nostro modello di traduzione usando alcuni semplici parametri fondati sui concetti di fertilità e distorsione che abbiamno appena illustrato:

- la *probabilità di fertilità* $P(n|e)$ cioé la probabilità che una data parola e abbia fertilità n

- la *probabilità di distorsione* $P(f|s.l)$ cioé la probabilità che una parola inglese che si trova nella posizione s corrisponda ad una parola francese che si trova in posizione f se la frase francese ha lunghezza l
- la *probabilità di traduzione* $P(f|e)$, una per ogni coppia di parole f (francese) e e (inglese) . Da non confondersi col caso in cui f ed e sono frasi!

All'inizio di questo discorso, avevamo descritto il modello di traduzione come un modo per calcolare (approssimare) $P(f|e)$, dove e è una frase inglese e f una francese. Adesso dobbiamo un po' rivedere quella definizione. Diremo, infatti, che il modello di traduzione è la probabilità $P(f.a|e)$ dove f è la solita frase francese, e è la solita frase inglese e a è uno specifico allineamento. La nostra probabilità, quindi, sarà quella che f sia la traduzione corretta di *e con un particolare allineamento a*.
Più avanti faremo considerazioni su cosa comporti questa nuova definizione per la nostra traduzione. Invece di esprimere formalmente la probabilità secondo questo nuovo modello di traduzione, vi mostrerò semplicemente come funziona per le due frasi d'esempio che avevamo visto in precedenza, con il rispettivo allineamento:

```
( Le chien est battu par Jean |
  John(6) does beat(3,4) the(1) dog(2) )
```

$$
\begin{array}{lll}
 & P(1|\text{John}) & \\
 & \times & P(\text{Jean}|\text{John}) \times P(6|1,6) \\
\times & P(0|\text{does}) & \\
\times & P(2|\text{beat}) & \\
 & \times & P(\text{est}|\text{beat}) \times P(3|3,6) \\
 & \times & P(\textit{battu}|\textit{beat}) \times P(4|3,6) \\
\times & P(1|\text{the}) & \\
 & \times & P(\text{Le}|\text{the}) \times P(1|5,6) \\
\times & P(1|\text{dog}) & \\
 & \times & P(\text{chien}|\text{dog}) \times P(2|6,6) \\
\times & P(1|\text{NULL}) & \\
 & \times & P(\text{par}|\text{NULL})
\end{array}
$$

La formula precedente dovrebbe risultare autoesplicativa, tranne l'ultima riga, che riguarda la parola francese "par". Questa parola non ha una prola corrispondente nella frase in inglese, cosa che abbiamo espresso con la parola speciale "NULL".

Diremo che il modello di traduzione è la probabilità $P(f.a|e)$ dove f è la solita frase francese, e è la solita frase inglese e a è uno specifico allineamento. La nostra probabilità, quindi, sarà quella che f sia la traduzione corretta di *e con un particolare allineamento a*.

Quel che rimane da fare è stimare i parametri impiegati per costruire questo modello: la fertilità, la distorsione e le probabilità della traduzione. Delineerò qui superficialmente una procedura iterativa che può essere impiegata per calcolare questa stima. Si comincia con una attribuzione di valori arbitraria a questi parametri. Per esempio potremmo dire che le probabilità di distorsione $P(f|s.l) = 1/l$ sono uniformi su tutta la frase. Simili assunzioni potrebbero essere adottate per gli altri parametri.
Per ogni coppia di frasi $(e.f)$ del nostro corpus possiamo impiegare il nostro assunto per calcolare la probabilità di tutti i possibili allineamenti fra le due frasi della coppia. Poi sceglieremo l'allineamento con probabilità maggiore come "il migliore".
Applicando questa procedura a tutte le frasi del corpus otterremo un allineamento *migliore* per ognuna. Potremo poi usare questa bae di allineamenti per calcolare anche gli altri parametri. Ad esempio se dovessimo trovare che un decimo degli allineamenti delle frasi di lunghezza 25 ha la prima parola mappata sulla prima parola della frase di destinazione, stabiliremo una nuova stima per $P(1|1.25)$ che sarà $P(1|1.25) = 1/10$. Questo costituisce una procedura applicabile iterativa-

mente per aggiornare i parametri del nostro modello; il processo potrà essere ripetuto molte volte.

Empiricamente troveremo (ma la cosa può anche essere dimostrata) che i parametri gradualmente convergeranno ad un punto fisso.

Cercare la soluzione che massimizza la nostra funzione

Avevamo delineato uno scenario nel quale la traduzione dal francese all'inglese potesse essere delineata come trovare una e che massimizzasse la funzione $P(f|e)P(e)$.

Adesso rivisitiamo il ragionamento che ci ha condotti a questa formulazione del problema e proponiamo una formulazione rivista che si adatti di più al nostro modello di traduzione fondato sugli allineamenti.

Invece di cercare una e che massimizzi $P(f|e)P(e)$, cerchiamo una coppia $(e.a)$ che massimizzi $P(e.a|f)$. Usando di nuovo il teorema di Bayes, possiamo riscrivere la nostra funzione $P(e.a|f)$ come

$$P(e.a|f) = \frac{P(f.a|e)P(e)}{P(f)}$$

Come in precedenza, siccome la f è fissa, possiamo omettere il denominatore $P(f)$ e il nostro problema diventa cercare una $(e.a)$ che massimizza

$$P(e.a|f) = P(f.a|e)P(e)$$

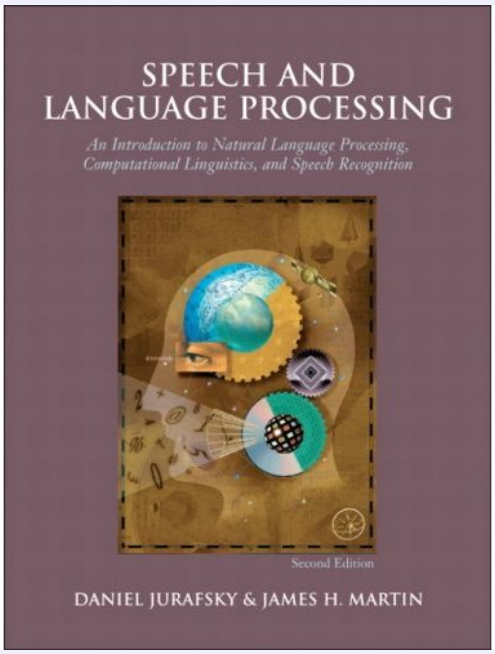
Ed ecco il nostro nuovo problema di traduzione: trovare una $(e.a)$ che massimizzi $P(f.a|e)P(e)$.

Sfortunatamente ci sono troppe possibilità perché la ricerca sia esaustiva, ma ci sono alcune euristiche che si comportano abbastanza bene. Una di queste è usare un algoritmo greedy che che gradualmente costruisce traduzioni parziali e allineamenti. Consideriamo le seguenti traduzioni parziali e i rispettivi allineamenti:

```
(Jean aime Marie | John (1) )
(Jean aime Marie | loves (2) )
(Jean aime Marie | Petroleum (1) )
```

Usando i modelli di lingua, di traduzione e di allineamento che abbiamo descritto, possiamo calcolare la probabilità di ognuno di questi allineamenti parziali e quindi calcolare il prodotto $P(f.a|e)P(e)$. Sarebbe davvero sorprendente se alle prime due di quelle frasi non venisse attribuità una probabilità elevata e alla terza una molto bassa.

Il libro Speech and Language Processing di Daniel Jurafsky, James H. Martin, è uno dei migliori riferimenti su questo argomento



La nostra euristica consisterebbe, dunque, nel tenere una serie di queste traduzioni parziali e allineamenti e poi continuamente estenderle finche non si realizzi una traduzione completa che abbia una probabilità significativamente più alta di quelle di tutte le traduzioni parziali.

> Invece di preoccuparsi se un certo algoritmo sia migliore di un altro, quelo che dovreste fare è prendere un insieme di dati di addestramento dieci volte più esteso. D'improvviso, l'algoritmo peggiore si comporterà meglio di quello che su un insieme più ristretto era il migliore. Preoccupatevi dei dati, prima che dell'algoritmo.

> Peter Norvig, direttore della Ricerca a Google, secondo quanto riferito da Greg Linden

Questo completa la nostra trattazione di un semplice sistema per la traduzione basato su una macchina statistica. Questo sistema è quello descritto nel 1990 da Brown ed altri presso i Laboratori di Ricerca IBM [2] ed è un sistema che traduceva in modo accettabile circa la metà delle frasi sottopostegli. Personalmente trovo davvero notevole il fatto che possano essere impiegate idee così semplici per fare progressi così grandi nella soluzione di un problema che si presenta come di enorme difficoltà (in articoli futuri parlerò di altri approcci allo stesso problema).

Inoltre, benché le idee di base siano rimaste le stesse, dal 1990 sono stati fatti ulteriori progressi, in questo campo. Il singolo avanzamento più importante è stato quello di muoversi da un modello che impiega le parole come unità findamentale ad uno che impiega intere frasi; il che ha aumentato considerevolmente la performance!

Se siete interessati a saperne di più, raccomando caldamente di consultare un documento dei Laboratori IBM del 1993 [3] che proseguiva le argomentazioni in merito e che introduce una panoplia di modelli di traduzione.

Mentre sul modello fondato su intere frasi non ho una singola citazione ma ho trovato una serie di voci fra le top hits di Google Scholar molto interessanti.

Infine, solo per gioco, vi raccomando l'eccellente introduzione al controllo ortografico (norvig.com/spell-correct.html) di Peter Norvig che impiega idee simili; in un contesto *molto* più semplice, ma col grande vantaggio di essere illustrato da pezzi di codice.

Riferimenti

1. La tesi di di dottorato di Philip Clarkson citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46
2. portal.acm.org/citation.cfm?id=92860
3. portal.acm.org/citation.cfm?id=972474
4. Su scholar.google.ca/ con le keyword "phrase based statistical machine translation"

`http://michaelnielsen.org`

Oltre il Knowledge Management, il Community Management

Una rivoluzione culturale in corso

Mina Distratis e Maurizio Mesenzani

Non molto tempo fa si parlava di una grande rivoluzione che avrebbe risolto molti problemi legati all'archiviazione delle informazioni e al loro riutilizzo. Si parlava di tecnologie dell'informazione: strumenti in grado di facilitare la gestione delle conoscenze e della documentazione. L'impatto è stato determinante da un punto di vista della mole di informazioni catalogabili, archiviabili e ricercabili ed ha comportato la necessità per le aziende di dotarsi di strumenti adeguati e di imparare ad utilizzare nuovi strumenti: gestori documentali, motori di ricerca, archivi elettronici, delegando a particolari figure (i Content Manager) la creazione e l'aggiornamento dei contenuti, implementando nuovi processi distribuiti di creazione e fruizione dei contenuti (a tal proposito conviene ricordare che il rapporto tra chi crea il contenuto e chi ne usufruisce è di 1 a 100). Ciò ha cambiato il rapporto tra le persone e le organizzazioni e il rapporto tra le persone e l'informazione.

Oggi infatti stiamo assistendo ad una vera rivoluzione, non solo tecnologica ma anche e soprattutto sociale e culturale. In primo piano non sono più le tecnologie e gli strumenti di gestione delle informazioni ma l'appartenenza ad una o più Comunità.

Appartenere significa condividere relazioni e accedere a una serie di informazioni aggiornate che si creano, si consumano e si trasformano nel corso della vita della community, grazie al continuo intensificarsi di relazioni e condivisione di risorse, oggetti, dati, strumenti... Informazione e persone diventano così inscindibili.

Chi progetta gli ambienti di suppporto sa che ora, oltre ad archiviazione, classificazione, gestione dei dati, reperimento e accesso veloce alle informazioni occorre pensare a relazioni, scambi, condivisione, collaborazione, flussi di informazioni.

Cambia profondamente il ruolo e il contributo delle persone: non più ricettori ma redattori, co-redattori, animatori, parti integranti dell'informazione. L'informazione senza le interazioni tra le persone perde valore, lo acquista proprio in virtù dei sistemi di relazione integrati con i contenuti: per esempio la fonte di un contributo, un commento o una valutazione di un esperto, il fatto di sapere che quell'oggetto è stato utilizzato da quella persona.

Il web 2.0 rende quindi reale il connubio persone–informazioni–relazioni.

L'appartenenza a reti sociali è ormai diventato un fenomeno di massa. Le rilevazioni fatte da ComScore mettono in evidenza che solo in Italia gli iscritti a Facebook sono 10 milioni e che nel mondo passano per facebook almeno 100 milioni di utenti al giorno. La persona diventa protagonista: come individuo è libera di pubblicare e di gestire le proprie informazioni; come membro di una Community si sente autorizzato a condividere, collaborare, prendere delle decisioni, informarsi, contattare...

L'individuo fa parte di vere e proprie reti sociali: appartiene a comunità multiple, anima le comunità, le vive, le popola di contenuti.

Questa rivoluzione socio-culturale in atto sta investendo anche le aziende in cui si fa largo una nuova idea di conoscenza che non risiede più nei database, ma si genera attraverso le Community ogni volta che le persone interagiscono in modalità reali o virtuali: incontri, workshop, convention, stanze on line, forum di discussione, blog. Qualunque sia il canale di contatto, le Community permettono generazione e scambio di conoscenze, rappresentando l'evoluzione naturale del knowledge management.

In tale contesto, l'obiettivo per le aziende diventa quello di facilitare le interazioni tra gli individui sostenendo la nascita e la valorizzazione di comunità e reti sociali. Si diffondono nuovi strumenti di:

- Comunicazione (es. le chat interne; le eConf ...)

in grado di favorire gli scambi informali, la responsabilizzazione, l'umanizzazione dei rapporti tra colleghi di diverso ruolo, funzione e area geografica

- Condivisione (es. l'eRoom, spazi virtuali strutturabili in base alle esigenze del gruppo e in cui il contenuto viene co-prodotto) che favoriscono l'intelligenza collettiva e la capitalizzazione
- Ricerca (es. Wiki integrati ai portali di KM per la co-produzione di contenuti e la ricerca per parole chiave definite dall'utente) che consentono di individuare velocemente "quello che sa", i documenti associati, temi e autori.

La distinzione tra chi crea e chi usufruisce dell'informazione crolla definitivamente e viene sostituita da partecipazione e co-produzione: si parla sempre più di redazione allargata, di flussi di content management distribuiti, di processi di knowledge management diffusi e pervasivi.
Ciò è vero anche nel rapporto dell'azienda con i suoi stakeholders:

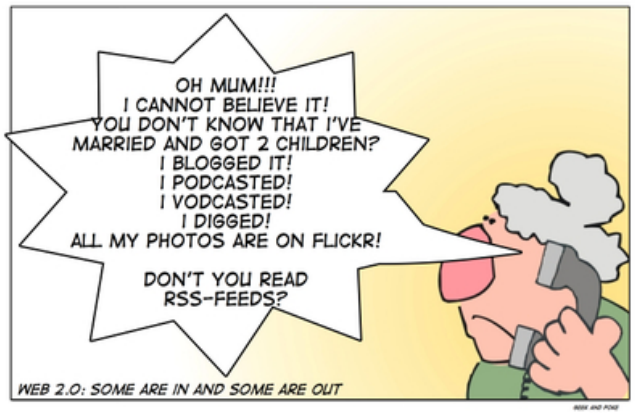
- Employees (B2E): collaboratori, dirigenti, parti sociali, candidati, ex dipendenti, alumni, ...
- Consumers (B2C): clienti, prospect, investitori, cittadini
- Business (B2B): fornitori, partner, Università, Istituti di Ricerca, Pubblica Amministrazione

Le tecnologie 2.0 consentono di gestire queste tre aggregazioni come community (o insiemi di community) e favoriscono quattro azioni sociali fondamentali:

- informare e dialogare (siti internet, intranet, blog, chat, forum, e-conf, ...)
- costruire e valorizzare le reti (siti internet, reti sociali pubbliche o private, strumenti di CRM, blog)
- favorire la collaborazione (rete sociale, wiki, opensource)
- condividere e innovare (siti internet, rete sociale, wiki, opensource)

La rivoluzione culturale è ancora in atto e alcune aziende hanno cominciato a muoversi adottando i primi strumenti di social networking e web 2.0, anche se diverse sono le resistenze da parte del management.
Da una parte la rilevanza degli investimenti tecnologici da sostenere: gli strumenti, seppur non particolarmente onerosi in sé, vanno integrati nelle architetture hardware e software esistenti, spesso obsolete o concepite in modo da centralizzare i processi di knowledge management e non per distribuirli
Dall'altra parte la necessità di cambiamenti spesso radicali ed integrati di tecnologie, organizzazione, processi e persone (in termini di skill, conoscenze e competenze). Quali gli scenari futuri? Quale la situazione generata dai cambiamenti in corso? Come sarà il knowledge management del futuro, quali strumenti sopravviveranno, quali tecnologie si diffonderanno e come cambieranno i rapporti tra organizzazioni, tecnologie e persone?
Le Community sono già una realtà: il futuro lo costruiremo a partire da qui.

Geek&Poke



Report dalla prima conferenza sul Software Libero in Kosovo

FLOSS in Kosovo

Flavia Marzano

L'Università di Pristina e l'associazione FLOSSK (Free Libre Open Source Software Kosovo www.flossk.org/) hanno organizzato la prima conferenza sul Software Libero in Kosovo www.kosovasoftwarefreedom.org/ che si è tenuta a Pristina dal 29 al 30 agosto presso la facoltà di Elettricità e Computer Engineering di Pristina.

Inaugurata da Rettore dell'Università, Prof.ssa Limani, la conferenza ha continuato con l'intervento del Vice Ministro Blerim Rexha, Professore alla Facoltà di Electrical and Computer Engineering Data and Security Network: un politico che parla la "lingua" del software libero, un politico che ha creduto in questa conferenza fin dall'inizio quando ancora non aveva forma né contenuti. L'ha sostenuta e cofinanziata e ha partecipato con un intervento sia tecnico che politico, cosa davvero più unica che rara.

Il programma è stato fitto e con moltissimi interventi da tutto il mondo (per la maggior parte con provenienza dai Balcani, ma anche interventi dall'MIT di Boston e da più di altri 10 paesi europeii) ha offerto un ventaglio completo su tutte le tematiche del software libero.

Dall'economia del dono agli applicativi più avanzati per la sicurezza e il VoIP, dalle licenze compresa la EUPL (descritta nella sua forma più giuridica dall'Avvocato PhD Giovanni Battista Gallus) al famoso One Laptop Per Child, dal Natural Language Processing al ruolo del software libero nella Pubblica Amministrazione e specialmente in un paese "nuovo" come il Kosovo.

Mike DuPont ideatore e organizzatore della conferenza ha avuto la pazienza, la costanza e l'energia di cercare ovunque tra i suoi contatti e con il concetto ormai adottato da molti del "viral marketing" relatori, keynote speakers, sponsor, patrocini e soprattutto una meravigliosa squadra di volontari, dai 14 ai 25 anni, che hanno collaborato a tutto, ma proprio a tutto e con una serietà e professionalità davvero encomiabile, tutto ha funzionato a perfezione in ogni sala, questi ragazzi hanno preso il loro incarico con una serietà da veri professionisti e sono stati i veri protagonisti dell'evento. Hanno fatto davvero di tutto: dall'andare a prendere all'aeroporto i vari ospiti alla gestione del coffee break, dalla gestione delle sale alla predisposizione del wifi per tutti, dalla definizione del programma alla definizione del logo, non un pinguino, ma un corvo: in Kosovo sono ovunque e passeggiano tra la gente.

L'intervento che ha avuto il maggiore successo è stato "Freedom, beyond Free of Charge" di Giuseppe Maxia che ha dichiarato: "La valuta dell'open source è il tempo, se non vuoi spendere soldi devi investire in tempo. Si può usare il software libero gratis perché tutte le informazioni e la documentazione relative sono disponibili gratuitamente, ma questo richiede tempo che viene poi trasformato in denaro e soluzioni."

"Per essere la prima edizione organizzata da volontari e probabilmente con un piccolo budget l'evento si rivela di estremo successo con grande affluenza di pubblico e interventi di grande rilievo", così ha detto Alessio Pennasilico di Alba S.T..

Una conferenza, ma anche una sorta di barcamp, aperto a chiunque volesse fare proposte.

La conferenza si è conclusa con la volontà di tutti di incontrarsi ancora l'anno prossimo e magari di far diventare questa conferenza non solo del Kosovo ma di tutti i Balcani, arrivederci allora nel 2010 in Kosovo terra di corvi e di software libero!

Rivoluzionaria idea della casa italiana in Brasile

La tua FIAT in CC?

FIATMIO.CC

Giovani Spagnolo

Michelangelo riteneva che la forma fosse già presente, in tutte le sue rifiniture, all'interno di un blocco di marmo, che ne fosse di esso prigioniera. Compito dell'artista era quello di estrarla e renderla visibile al pubblico.

È un po' quest'idea che ha guidato la FIAT quando ha lanciato in Brasile il progetto FIATmio, una Concept Car che, prima al mondo, è rilasciata con licenza Creative Commons.

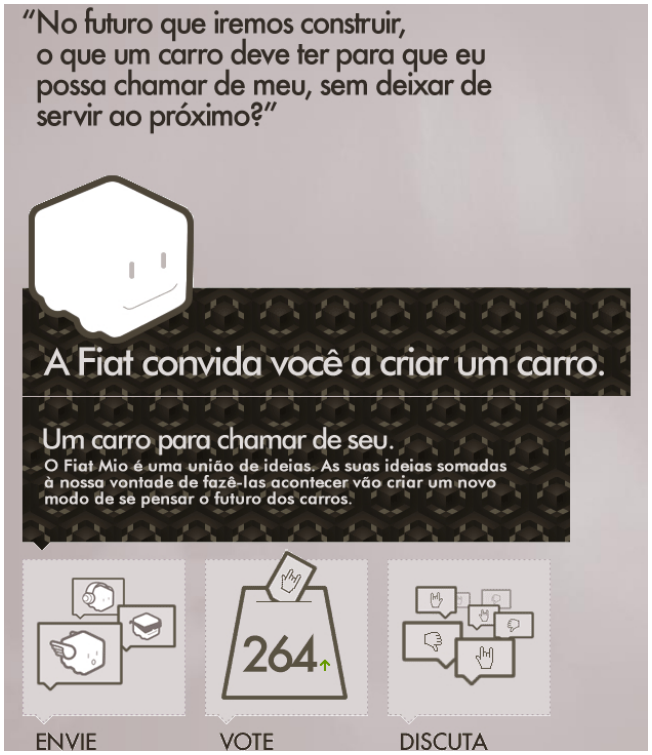
FIATmio è una lodevole iniziativa di *crowdsourcing* per coordinare i suggerimenti di chiunque voglia partecipare alla costruzione delle prossime generazioni di auto e raggrupparli in un grande "blocco di idee" che diverrà punto di partenza per i futuri sviluppi del mercato automobilistico.

Sul sito www.fiatmio.cc (attualmente solo in portoghese purtroppo) gli utenti possono proporre idee di progettazione per l'auto in diverse categorie tra le quali Propulsione (come la macchina si sposterà), Infotainment (come la macchina si comunicherà con l'utente), Design, Materiali, Sicurezza, ed altre... Tutti possono votare le migliori idee e lasciare il loro commento. È possibile inoltre leggere le idee più interessanti seguendo l'account Twitter @fiatmio.

La Fiat ha adottato la liberale licenza "Creative Commons Attribuzione-Condividi allo stesso modo", nella sua versione brasiliana 2.5, per poter aggregare e diffondere le idee degli utenti attraverso il sito e successivamente utilizzarle all'interno dell'azienda. Anche tra le licenze CC è una scelta lungimirante.

Il team brasiliano di ingegneri automobilistici realizzerà una serie di test di fattibilità e produrrà il primo concetto, la Fiat Concept Car III (FCC III), con l'obiettivo di presentarla al Salone Internazionale dell'Automobile nel 2010.

Nel sito fiatmio.cc, la stessa Fiat dice che "è importante far sapere che tutto il contenuto sarà libero. La Fiat ritiene che la conoscenza generata in questo progetto debba essere diffusa senza restrizioni e che possa essere utilizzata non solo da singoli utenti ma anche da ingegneri e altri produttori di veicoli."



La Fiat non ha spiegato perché il progetto è stato lanciato prima in Brasile, ma si può immaginare che con la presenza di una sempre più attiva comunità di Software Libero e con l'85% degli utenti web iscritto e utilizzando attivamente almeno un sito di social network, il paese sia il posto perfetto per sperimentare nuovi modi di innovare in un mercato globale sempre più competitivo.

Ai suoi tempi il maestro Michelangelo non sarebbe del tutto d'accordo, ma c'è senza dubbio da sperare che presto anche gli italiani possano creare la loro macchina dei sogni attraverso il Web.

Divertimento, passione e professionalità

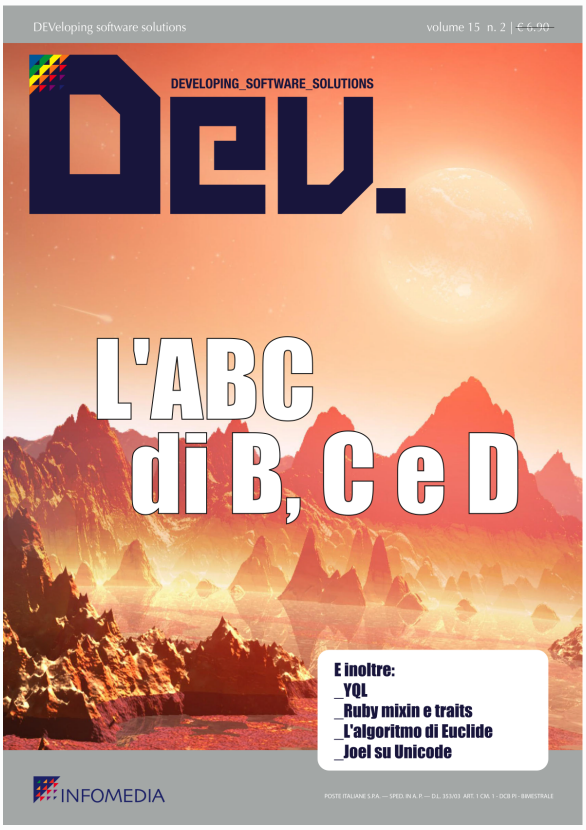
Computer Programming

Computer Programming è la più antica e di gran lunga la più conosciuta rivista di programmazione in Italia.

Fortemente impegnata a fornire informazioni di qualità in materia di sviluppo e di programmazione, dal 1991 Computer Programming è il 'salva-vita' di riferimento per gli sviluppatori italiani grazie alla sua indipendenza, alla qualità degli articoli, all'acutezza di analisi, agli autori di prestigio e all'attenzione alle esigenze dei lettori. Su Computer Programming, si possono anche trovare interviste a personaggi che realizzano e scrivono l'informatica moderna come Wirth, Stroustrup, Meyer, Waldo, Gaslin, Stepanov, Brockschmidt, Ritcher. Ogni numero di CP è tutto questo: idee, soluzioni, competenza e professionalità a 360 gradi.

La trovi su
www.infomedia.it/i/cp

DEV



DEV è la rivista di programmazione per tutti.

DEV permette di iniziare la programmazione dall'ABC.

DEV è la fonte per i principianti con cui si può approcciare questo affascinante mondo, con articoli introduttivi e corsi di programmazione. Ma non solo!

È un mix di tecniche, trucchi, consigli e soluzioni anche per gli esperti che desiderano rimanere aggiornati.

Linux, Windows, Visual Basic, Python, C/C++, Java, .NET, programmazione Web, LAMP, SQL, computer grafica sono solo alcuni degli argomenti trattati che rendono DEV la più appetitosa rivista italiana per gli sviluppatori e i praticoni.

Consultala su
www.infomedia.it/i/dev

L'ecosistema Infomedia

Turing Club

Il *Turing Club* è la prima (e unica) associazione dei programmatori professionisti, e appassionati di programmazione in Italia.

Ci ispiriamo ad Alan Mathison Turing, perché tra i padri della computer science è stato il primo a capire che la nascita del calcolatore digitale avrebbe fatto sorgere una nuova leva di professionisti e che si sarebbe sviluppata una loro cultura e professionalità specifica. Turing realizzò il primo linguaggio e fu quindi senza dubbio il primo programmatore in senso moderno.

Poiché troviamo che la figura di Turing, in Italia, sia ricordata solo legata alla vasta aneddotica sulla sua vita turbolenta, o un po' per i suoi importanti contributi scientifici, ma troppo poco rispetto al suo ruolo di innovatore delle strutture sociali e culturali, noi programmatori e sistemisti vogliamo segnare, con questa attribuzione, il ringraziamento imperituro a questa fondamentale figura anche della nostra cultura nella quotidianità della nostra vita.

Il Turing Club tiene alla promozione della cultura informatica, specie tra le giovani generazioni e per questo si impegna. L'informatica ha infatti cambiato la vita della nostra Società, ma troppo spesso si immagina che alle spalle di questa innovazione ci sia una sorta di incomprensibile magia. Questa magia non è che il frutto di giorni e notti di programmazione appassionata, il cui valore sociale è ben superiore al guadagno che se ne ricava.

In una società come quella italiana poi, per ragioni anagrafiche e generazionali, per l'assenza di un sistema di credito avanzato e per la miopia della classe direttiva e politica, il talento, la competenza e l'innovatività dei programmatori piuttosto che essere premiata, viene giorno dopo giorno disconosciuta e marginalizzata, con il pratico risultato di spingere l'Italia nel Medioevo tecnologico.

Chiunque tu sia, quale che siano i tuoi strumenti di lavoro, il tuo linguaggio di programmazione, la tua piattaforma o framework, il tuo sito o il tuo programma... questo è il Turing Club

Il Turing Club offre molte possibilità di partecipazione nelle attività associative in forma volontaria e non retribuita. Partecipando è possibile sfidare le proprie competenze, dimostrare liberamente le proprie abilità, migliorare la propria carriera e rendere migliore l'ambiente informatico italiano.

Il Turing Club riconosce e promuove l'attività in campo informatico indipendentemente dalla partecipazione diretta all'associazione. Comunque crediamo fermamente che l'attività volontaria sia una chiave di successo vitale per la comunità informatica italiana ed è nostro specifico obiettivo riconoscerla ed incentivarla.

L'informazione tecnica, sull'esempio delle grandi società professionali estere, è realizzata dagli esperti della comunità e non prevede alcuna forma di remunerazione. I volontari aiutano l'associazione ad organizzare, pianificare e produrre tutti i nostri servizi, i nostri periodici, le pubblicazioni e le conferenze, per puro spirito di servizio e partecipazione alla comunità nazionale.

Per maggiori informazioni
volontari@programmers.net

http://the.programmers.net

Infomedia Editori Società Cooperativa

Infomedia è l'impresa editoriale che da quasi venti anni ha raccolto la voce dei programmatori, dei sistemisti, dei professionisti, degli studenti, dei ricercatori e dei professori d'informatica italiani.

Sono più di 800 gli autori che hanno realizzato per le testate Computer Programming, Dev, Login, Visual Basic Journal e Java Journal, molte migliaia di articoli tecnici, presentazioni di prodotti, tecnologie, protocolli, strumenti di lavoro, tecniche di sviluppo e semplici trucchi e stratagemmi. Oltre 6 milioni di copie distribuite, trentamila pagine stampate, fanno di questa impresa la più grande ed influente realtà dell'editoria specializzata nel campo della programmazione e della sistemistica.

In tutti questi anni le riviste hanno vissuto della passione di tanti, come te, che vedono nella programmazione non solo una professione ma una passione vitale e un vero divertimento.

Mettere in condivisione le soluzioni adottate nello sviluppo di un programma, ripercorrendone le tappe è un ottimo modo per documentare al meglio il proprio lavoro, fissare i concetti e verificare la qualità delle proprie realizzazioni. È uno splendido modo per dare agli altri la possibilità di impare nuove tecniche e progredire nelle proprie competenze e per rendere il nostro ambiente più vitale, competitivo ed attivo.

La programmazione o la soluzione dei problemi sistemistici ci dà molta gioia, perché è necessario superare i limiti della nostra conoscenza. A volte abbiamo la sensazione che non è possibile comunicare tale piacere. Questo è il luogo giusto per farlo. I tuoi pari sono interessati alle tue soluzioni, ai passi che hai compiuto per raggiungere un risultato e perché no, agli errori che hai commesso nei tuoi tentativi.

L'informatica è un ambito professionale unico, in cui le spinte competitive sono sempre state forti, ma altrettanto forti ed importanti è stata la tensione a cooperare. La condivisione delle informazioni e la diffusione delle conoscenze sono l'anima costitutiva di questa Comunità.

Le nuove Computer Programming, Dev e Login vogliono incentivare questi aspetti per essere ancora di più al centro di questa Comunità. Aiutateci. Perché senza queste riviste, programmatori e sistemisti italiani rimarrebbero senza voce.

Chiunque tu sia, quindi, quale che sia la tua storia, i tuoi strumenti di lavoro, il tuo linguaggio di programmazione, la tua piattaforma o framework, il tuo sito o il tuo programma... contribuisci a questa Storia per permetterci di andare avanti.

Sito: http://www.infomedia.it
Blog: http://blog.infomedia.it
Wiki: http://wiki.infomedia.it

Sede Legale: Via Iannacchini 28, 83100 Avellino
Partita Iva: 02564790646
N. Tel.: 0289605440

InformAzioni

ED-BOARD

Per accedere all'`ed-board` di Infomedia si può aderire alla mailing-list http://groups.google.it/group/infomedia-edboard. All'interno di questa mailing-list si discutono gli argomenti e linee evolutive per le riviste.

Il lavoro dell'`ed-board` ha una prospettiva semestrale e ciclo trimestrale.

La prossima chiusura della mailing-list è quindi prevista per il 30 Settembre 2009. A quella data tutte le adesioni alla mailing-list (dopo un avviso) verranno cancellate. Chi vorrà potrà liberamente reiscriversi per partecipare al ciclo trimestrale successivo, la cancellazione automatica è una regola che tende a mantenere in lista solo persone effettivamente attive. (ovviamente è possibile cancellarsi da soli in qualsiasi momento se non si è più interessati).

Il semestre di riferimento per l'attuale `ed-board` è quindi quello che inizia alla scadenza di questo trimestre, ovvero 2009Q4 e 2010Q1.

I trimestri precedenti (2009Q2 e 2009Q3) non hanno avuto il supporto di un `ed-board`, poiché l'istituzione di quest'organo, in questa forma, risale alla metà di Maggio del 2009.

La presenza nell'`ed-board` comporta l'iscrizione del nome dei partecipanti, se si vuole, all'interno del colophon della rivista.

PER INFORMAZIONI
GROUPS.GOOGLE.IT/GROUP/INFOMEDIA-EDBOARD

EDITORIAL-BOARD

I partecipanti all'`edboard` del 2–3 trimestre 2009 (2009Q2-2009Q3) sono:

Aaron Brancotti (aka Babele Dunnit)
Adriano Peluso
Andrea Baruzzo
Andrea De Carolis
Davide *folletto* Casali
Dimitri *megadix* De Franciscis
Emmanuele Somma
Gianluigi Spagnuolo
Giovanni Dall'Olio
Ignazio Licata
Luca Missori
Maurizio *.mau.* Codogno
Max Arcidiacono
Orazio Del Franco

LO STATO ATTUALE

Fare o non fare certe cose dipende dalla disponibilità di volontari, questo è il nostro nuovo modello basato sul volontariato professionale.

È un modello mutuato dalle migliori esperienze nel campo, che può senza dubbio crescere grazie all'impegno di quanti vorranno darci concretamente una mano. Quale che sia il tuo impegno, piccolo o grande, sarà importante per noi e, crediamo, per la nostra professione.

Il processo di lavoro che è al momento attivo, altre alla comunicazione e alle informazioni per la community, è quello redazionale che permette la pubblicazione online delle due testate residue in forma bimestrale.

È un processo abbastanza rodato che potremmo considerare ad uno stato *inziale* di maturità, per quanto risenta della mancanza di risorse umane: siamo infatti lontani dal target dei quattro editor (part-time) che richiederebbe per stabilizzarsi.

Non siamo ancora in grado di conteggiare specificamente l'impegno necessario per la realizzazione completa della rivista ma la sensazione che si desume dal lavoro di questi ultimi mesi è di circa 4 giorni/uomo per numero, il che con il target di quattro editor porterebbe l'impegno personale di un editor a circa 10 ore al mese. Un coinvolgimento, crediamo, ragionevole per un'attività volontaria.

Il vero problema è che per 'entrare' in questo processo è necessario avere un insieme di competenze nient'affatto banali.

Vale la pena però di sottolineare che attualmente l'impegno da parte dei due editor attuali è alquanto superiore, in parte perchè sono stati 'beta-tester' del software realizzato per l'impaginazione in parte perché hanno anche assolto ad una funzione di progettazione ed ideazione delle riviste, nonché talvolta alla traduzione/redazione di articoli.

Non possiamo che rinnovare l'invito alla partecipazione personale. È una esperienza importante e significativa.

SE VUOI PARTECIPARE COME EDITOR ALLA REALIZZAZIONE DELLE RIVISTE INFOMEDIA SCRIVI A **REDAZIONE@INFOMEDIA.IT**

Editor: Cosa fa?

L'editor nella struttura Infomedia assolve essenzialmente a tre compiti, privilegiandone l'uno o l'altro secondo la propria natura:

progetta i numeri correnti delle riviste seguendo le discussioni e le indicazioni che sorgono all'interno dell'`ed-board` e con il materiale disponibile nell'inbound;

contatta gli autori in base alle loro competenze o attitudini, li segue nella redazione degli articoli stimolandoli a rispettare le linee guida e i tempi di redazione, verifica la disponibilità del materiale ed eventualmente provvede alle necessarie sostituzioni;

trasforma, se necessario, il materiale ottenuto dagli autori in file di produzione per il sistema di impaginazione automatizzata TCHL, verifica la presenza di tutti gli elementi per una corretta impaginazione (titoli, didascalie, immagini, ecc. ecc.), completa gli articoli, se necessario, con opportune integrazioni (box informativi o riferimenti).

Le prime due attività sono essenzialmente relazionali, per lo più condotte utilizzando l'email, i gruppi di discussione o la chat, mentre l'ultima di natura sostanzialmente tecnico-operativa, e si svolge con l'utilizzo dell'insieme di strumenti sviluppati appositamente dal Turing Club e che prevedono l'utilizzo di LaTeX, Emacs, Python e Subversion.

Per informazioni
www.assembla.com/wiki/show/turing

Traduttori

Legate al processo redazionale sono le traduzioni. Le nuove riviste si stanno caratterizzando anche per la notevole apertura al dibattito internazionale sulla programmazione, e per la disponbilità di interventi di autori di grande calibro.

La prospettiva delle traduzioni è interessante per due motivi. Innanzitutto esistono molti articoli che la 'cultura informatica' italiana conosce poco e la cui divulgazione diventa un importante occasione di evoluzione culturale. Poi la traduzione può essere un compito non così impegnativo per chi vuole aiutarci ma allo stesso tempo sfidante, facile da realizzare in comunità e interessante da revisionare. La mailing-list dei traduttori http://groups.google.it/group/infomedia-translate ha già cinque membri attivi.

Manca però un 'coordinatore' a cui si possa delegare il rapporto con i traduttori, una operazione essenzialmente gestionale che permetta di garantire ai redattori l'arrivo nei tempi stabiliti degli articoli. Il coordinatore dovrebbe stabilire e verificare le agende di traduzione (avvertendo per tempo la redazione di eventuali intoppi), e se possibile revisionare (o coordinare la revisione) delle traduzioni. Si tratta di farlo per un paio di articoli al mese. L'impegno mensile del coordinatore non dovrebbe essere lontano, tutto sommato, da una 'giornata-uomo', come per l'editor ma le competenze necessarie sono senza dubbio meno specialistiche (linguistiche e relazionali).

Per lavorare negl gruppo dei traduttori
aderisci e presentati nella mailing-list
groups.google.it/group/infomedia-translate

TODO

Esistono alcuni processi, a loro modo 'secondari' che rimangono indietro, sui quali agiamo in modo sporadico. Molti. Troppi anche solo per farne un censimento. Il più importante dei quali è la prospettiva di riattivare il processo di stampa che però ha bisogno di un supporto professionale che in questo momento ci è venuto meno.

Il secondo è quello, importante, di un 'abbozzo' almeno di 'servizio clienti' che risponda alle email amministrative (su questo qualcosa si sta facendo in 'batch').

Infine la gestione dei servizi web, che necessitano ampiamente di sostegno e supporto.

Un aiuto in questi campi è molto gradito.

Se ti interessa partecipare a qualcuno
di questi processi consulta le pagine relative
su **wiki.infomedia.it**
e contatta **amministrazione@infomedia.it**

Ed Inoltre

Questa rivista non potrebbe essere stata realizzata senza la disponibilità e il lavoro di tutte le persone a vario titolo coinvolte.

Un ringraziamento particolare va anche a:

Adele Picardo
Francesco Benevento
Daniela Della Porta
Dino Meoni
Donato Marcello Sabato
Gianluca Pignalberi
Gianluigi Coccia
Luciano Talarico
Maria Letizia Mari
Natale Fino
Noemi Somma

"Non cambierai mai le cose combattendo la realtà esistente. Per cambiare qualcosa, costruisci un modello nuovo che renda la realtà obsoleta." **R. Buckminster Fuller**

Gli autori di questo numero

Michael Nielsen

si è formato alla Università del Queensland e come studente Fulbright Scholar all'Università del New Mexico. Ha lavorato al Los Alamos National Laboratory, come Richard Chace Tolman Prize Fellow del Caltech, è stato Foundation Professor di Informatica Quantistica e Federation Fellow all'Università del Queensland e Senior Faculty Member all'Istituto Perimeter per la Fisica Teorica. Ha lasciato l'istituto Perimeter per scrivere un libro sul futuro della scienza.

Email mn@michaelnielsen.org
Blog http://michaelnielsen.org

David *folletto* Casali

Interaction Designer, fondendo differenti ambiti disciplinari quali il design, la programmazione, la ricerca psicosociale e la grafica rappresenta una figura professionale ibrida, seguendo i principi di sintesi, semplicità e naturalezza. Lavora in Maison,the e partecipa in progetti quali WideTag.com, Good50x70.org, Tonight.eu, Bzaar.net e Idearium.org.

Blog im.digitalhymn.com

Giovani Spagnolo

è consulente informatico e Direttore di Programmi della Telematics Freedom Foundation

Twitter giovani
Identi.ca giovani

Marta Veneri

si è laureata in matematica presso la Facoltà di scienze matematiche, fisiche e naturali dell'Università di Siena. Lavora, dal febbraio 2003, per Etruria Innovazione Scpa, è JAVA Senior Developer ed attualmente si occupa di progettazione di piattaforme per i sistemi museali.

Maurizio Mesenzani

è Amministratore Delegato di arké management consulting services, società di consulenza, formazione, sviluppo tecnologico e servizi alle persone nata nel 2008 da un gruppo di professionisti esperti che operano in rete tra Milano, Firenze, Roma e Rio de Janeiro. Lavora con clienti pubblici e privati tra cui Banca Mediolanum, Banca Sella, Vodafone, ING Direct, Provincia di Roma, Ancitel. Dal 2007 collabora con Fondazione Irso, dopo numerosi anni di attività svolte con l'istituto di ricerca Irso, creato da Federico Butera. E' tra i partner di Fondazione Irso e Responsabile del Customer Management Forum. Ha svolto il ruolo di Direttore di Produzione in TILS spa (Telecom Italia Learning Services) con responsabilità della struttura di consulenti e formatori (130 persone) che comprendeva la Scuola di Formazione Reiss Romoli dell'Aquila. Ha svolto attività di consulenza su organizzazione e management presso Butera e Partners. Ha gestito interventi di progettazione strategica ed organizzativa, formazione e sviluppo tecnologico in Vodafone, H3G, ING Direct, Poste Italiane e in Monte Titoli (Gruppo Borsa Italiana). Ha lavorato in Andersen Consulting, nell'area di competenza Process. In Andersen Consulting ha ricoperto il ruolo di Process Excellence Competency Knowledge Manager per l'area EMEAI (Europa, Middle East e Sud Africa), ha partecipato allo sviluppo della linea di offerta sull'e-Business e ha partecipato alla realizzazione di progetti di Knowledge Management presso aziende di produzione industriale. In precedenza ha svolto attività di ricercatore, consulente e formatore sui temi connessi all'organizzazione del lavoro e alla gestione per processi in aziende pubbliche e private collaborando con Irso Srl e RSO spa. Collabora con enti professionali e istituzioni accademiche in attività di docenza e gruppi di studio su organizzazione e management. Sugli stessi temi ha pubblicato articoli e contributi su riviste specializzate. Professore a contratto presso l'Università Bicocca di Milano, laurea specialistica in sociologia, materia Comunicazione e Management dei Servizi.

Email mesenzani@libero.it

Ron Burkey

per un breve periodo è stato ricercatore di Fisica, ma la gran parte della sua carriera professionale l'ha trascorsa nell'industria progettando hardware embedded con il relativo firmware, gran parte dei quali per impiego aereonautico. Recentemente ha scoperto le meraviglie del software open-source nella creazione e distribuzione del quale si è lasciato coinvolgere per hobby.

Sito http://www.sandroid.org

Francesco Bini

è responsabile allo sviluppo di Etruria Innovazione SCpA. Si occupa di sistemi distribuiti e sistemi real-time con particolare attenzione verso algoritmi di Intelligenza Artificiale nel campo della visione e del linguaggio naturale. E' laureato in Ingegneria delle Telecomunicazioni.

Email francesco.bini@etinnova.it

Giacomo Veneri

è dottorando presso l'Università degli Studi di Siena dipartimento di scienze Neurologiche e del Comportamento. Si occupa di sistemi di Eye-tracking e modellazione del sistema nervoso centrale con particolare riferimento alla visione.

Email g.veneri@unisi.it

Luca Colucci

è attualmente .NET Senior Developer presso Etruria Innovazione Scpa di Siena. È laureato in Ingegneria delle Telecomunicazioni presso l'Università degli Studi di Siena e ha svolto dal 2004 al 2006 la professione di consulente informatico presso per la Capgemini Media & Networks Italia S. p. A.

Mina Distratis

è una sociologa delle organizzazioni, dal 2001 è consulente e ricercatrice sui temi del KM, della tecnologia integrata al sistema sociale, degli spazi di lavoro, dell'analisi e progettazione organizzativa. Dal 2009 è socio di amcservices, società di consulenza, formazione, sviluppo tecnologico e servizi alle persone. Segue per conto di Renault S.p.A. i progetti BtE, di KM e di miglioramento dei processi. Attualmente lavora su progetti di "Comunicazione unificata" e "Collaborazione a distanza". Fa parte dell'Osservatorio Francese Web 2.0. Per Renault Consulting s.a.s., ha partecipato alla ricerca sugli impatti delle nuove tecnologie sul lavoro collaborativo. Collabora con Cesim su KM e portali intranet. Ha svolto attività di consulenza e ricerca su organizzazione e processi per conto di Butera e Partners e Irso. Ha pubblicato con la Franco Angeli: *Team per il successo. Riconoscere, progettare e gestire team autoregolati ad alta performance*, De Bernardis, Distratis, Ferrari, Nunziata, 2007 e *Creare Contesti per Innovare: la dimensione fisica dell'innovazione*, Distratis, Ferrari, Paoli, 2006

Email mina.distratis@amcservices.it
Sito www.amcservices.it

Pamela Federighi

è laureata in Ingegneria Elettronica presso l'Università degli Studi di Pisa, Italia.

Attualmente svolge attività di ricerca presso il Dipartimento di Scienze Neurologiche, Neurochirurgiche e del Comportamento presso l'Università degli Studi di Siena. I suoi interessi di ricerca riguardano l'analisi dei segnali biomedici, le applicazioni mediche e la bioinformatica.

Collabora con aziende impiegate nel settore della ricerca e della consulenza riguardo a nuovi materiali e nuove tecnologie per differenti campi di applicazione.

Flavia Marzano

è laureata in Scienze dell'Informazione all'Università di Pisa e da più di venti anni operativa nel management in settori di punta ha come obiettivo primario quello di rispondere alla necessità di pianificazione dell'Innovazione della Pubblica Amministrazione a partire dalla ristrutturazione e dal riequilibrio dei Sistemi Organizzativo, Informativo ed Informatico. Docente di Scenari e Innovazione delle Tecnologie dell'Informazione (Università Bologna), membro della Commissione nazionale sul software libero (Ministro Innovazione e Riforme nella PA), valutatore esperto per la Commissione Europea (FP6), membro del Comitato scientifico del Master in Management OSS (Università Pisa), membro del Comitato Scientifico FLOSS Provincia Roma, membro del Centro Regionale di Competenza per il Riuso (Toscana), Collabora con il Dipartimento Affari Regionali per il Programma ELISA.

Email flavia.marzano@gmail.com
Sito finalmentelibero.ning.com
Blog undo.ilcannocchiale.it

Googletoon

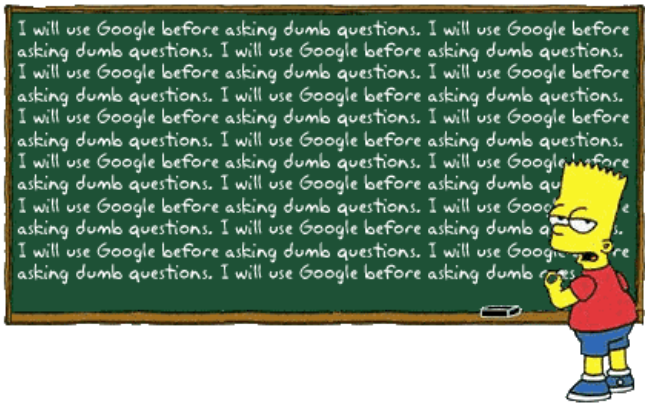


Photo Credit: NASA

La Luna ripresa dal modulo di comando dell'Apollo 12 mentre lasciava l'orbita lunare (11/24/69).

Le altre immagini:

Le vignette a pag. 54 sono tratte dal web-comic Geek&Poke di `http://geekandpoke.typepad.com/`, mentre quella di pag. 13 è XKCD di Randall Munroe `http://xkcd.com/`

Lo sfondo dei titoli dello speciale rappresenta il modulo lunare LEM `http://www.nasa.gov/centers/goddard/images/content/238403main\_moon.jpg`

La redazione ha compiuto tutti gli sforzi per accreditare opportunamente il copyright delle immagini presentate.
Per eventuali errori, che saranno immediatamente corretti, scrivere a redazione@infomedia.it

Proponi il tuo articolo

Se vuoi diventare un autore per le riviste Infomedia i passi da seguire sono:

Fase di progettazione

In questa fase è importante rendere nota alla redazione la propria intenzione di scrivere un articolo e dettagliarne le caratteristiche.

Per farlo è essenziale sempre compilare il modulo disponibile a questo indirizzo:

http://mags.programmers.net/proposta-articoli

Attenzione, indica sempre una data di consegna e una email valida (valuteremo il tempo di cui avrai effettivamente bisogno per realizzare l'articolo considerando la data d'inserimento del modulo, ovvero se il modulo viene inserito il 1 Maggio e vi si indica come data di consegna il 15 Maggio noi considereremo che, in qualsiasi momento dovremmo mai chiedere l'articolo la sua redazione potrebbe portar via all'autore una quindicina di giorni, ovviamente questa cosa sarà confermata nel contatto successivo).

In redazione usiamo questo modulo quando facciamo la programmazione di dettaglio dei vari numeri.

Visto che ti trovi compila subito anche la manleva (per stabilire la licenza di cessione dell'articolo a Infomedia, vedi box Licenza di distribuzione)

http://mags.programmers.net/moduli/manleva

e la biografia, che sarà riportata sulla rivista e sul sito:

http://mags.programmers.net/moduli/biografia

A questo punto, indipendentemente dall'accettazione dell'articolo, ti consigliamo di dare una lettura, anche superficiale alle linee-guida. Sebbene attualmente non siano da considerare requisiti stringenti è importante entrare nel 'mood' del nostro sistema di open-publishing. Purtroppo anche questa documentazione non è completa (ma ce ne è già a sufficienza, però).

http://mags.programmers.net/mags/linee-guida

Redazione

Quando la redazione ti contatterà per chiederti se vuoi veramente scrivere l'articolo e per confermare i tempi di consegna ti indicherà anche l'effettivo numero di pagine da riempire.

Cerca di mantenerti dentro le dimensioni assegnate o non appena verifichi di aver bisogno di più pagine informane immediatamente la redazione.

Se invece 'arrivi corto', provvedi in autonomia ad aggiungere elementi opzionali (figure, riquadri con recensioni o biografie,o elementi 'zoom').

Imparando ad utilizzare gli strumenti di open-publishing è possibile fare in autonomia una impaginazione preliminare per verificare come l'articolo potrebbe venir impaginato e comprenderne gli eventuali problemi (figure troppo piccole, pezzi di codice troppo ampio, ecc.)

Stiamo studiando una modalità di consegna degli articoli direttamente in Subversion, per ora zippa tutto e mandalo a

REDAZIONE@INFOMEDIA.IT